El legado olvidado

Autor: Enrique Hernández Hernández
Tutor: Jordi Duch Galvadá
Profesor: Joan Arnedo Moreno

Máster en Diseño y Programación de Videojuegos
Programación avanzada

18 de junio de 2023

Créditos/Copyright

FICHA DEL TRABAJO FINAL

Título del trabajo:	*El legado olvidado*
Nombre del autor:	*Enrique Hernández Hernández*
Nombre del colaborador/a docente :	Jordi Duch Gavaldá
Nombre del PRA:	Joan Arnedo Moreno
Fecha de entrega (mm/aaaa):	*06/2023*
Titulación o programa:	*Máster universitario de Diseño y Programación de Videojuegos*
Área del Trabajo Final:	*Programación Avanzada*
Idioma del trabajo:	*Español*
Palabras clave	*Videojuego, aventura gráfica*

Resumen del Trabajo (máximo 250 palabras):

Nivel de un juego en el personaje se moverá con teclado o un joystick y la cámara con el ratón o con otro joystick. A la vez se dispondrá de un inventario que almacenará los objetos que recoja por la escena. Estos objetos recogidos se podrán usar con otros interactuables o personajes para poder desbloquear nuevas zonas, diálogos u objetos que estaban ocultos anteriormente. Todo ello llevará a resolver retos propuestos que permitan avanzar en la historia y la aventura. Mezclado con la aventura gráfica se busca introducir zonas de sigilo en las que debemos sobrepasar a enemigos que estén vigilando o incluso acabar con ellos de manera sigilosa.

Referencias del juego pueden ser Broken Sword 3 y 4, El Código Da Vinci o A Plague Tale, en los que se proponen retos intelectuales al jugador mezclado mediante retos o puzles a la vez que hay ciertas zonas de sigilo y más dinámicas.

El objetivo del trabajo es mediante la herramienta Unity y el lenguaje C# crear un nivel totalmente jugable con todas las mecánicas de movimiento, diálogos, inventario y recolección de objetos y puzles como si de una aventura gráfica se tratara. Además de incluir una zona de sigilo como la nombrada anteriormente que se deba recorrer sin ser visto y acabar con algún enemigo en silencio.

Abstract (in English, 250 words or less):

Level of a game in which the character will move with the keyboard or a joystick, and the camera with the mouse or another joystick. At the same time, there will be an inventory that will keep the objects that you pick up in the scene. These collected objects can be used with others interactive or characters to unlock new areas, dialogues or objects that were previously hidden. All of this will lead to solve proposed challenges that allow the story and adventure to progress. Mixed with the graphic adventure, the goal is to introduce stealth areas in which we must overcome the enemies that are watching or even kill them silently.

Some references of the game can be Broken Sword 3 and 4, The Da Vinci Code or A Plague Tale, in which intellectual challenges are proposed to the player, mixed with challenges or puzzles, at the same time that there are certain areas of stealth and more dynamic.

The purpose of the thesis is to use the Unity tool and the C# language to create a fully playable level with all the mechanics of movement, dialogue, inventory and object collection, and puzzles, as if it was an adventure game. In addition to including a stealth area like the one mentioned above that must be traversed without being seen and finish off an enemy quietly.

Cita

“Mi principal queja sobre los videojuegos es que no son lo suficientemente buenos para los adultos.

Para que un adulto disfrute algo, necesita de estímulos intelectuales, algo relacionado a la vida real. Jugar póker enseña a engañar a la gente, algo que tiene relevancia en la vida real… pero un tiro en la cabeza con un rifle francotirador carece de importancia. Los juegos tienen que ser intelectualmente relevantes.

También necesitas profundidad: está la aventura, la emoción de vivirla, pero buscas que te ponga la piel de gallina.

¿Pueden los juegos enseñar algo intelectual y relevante de y entre dos personas? ¿Pueden los juegos hacer que 2 personas experimenten algo emocional y profundo que los afecte como adultos? En eso estoy trabajando, en hacer juegos emocionales e intelectualmente relevantes. En hacer juegos en los que la gente se pueda conectar con otros y pueda reunirse”.

Jenova Chen, creador de Journey

Agradecimientos

En primer lugar, quiero agradecer a mi familia por todo el apoyo y empuje que me han dado a lo largo de estos años para conseguir llegar a todas mis metas académicas y vitales. A mi hermana por transmitirme esa fuerza de voluntad que la caracteriza, a mis padres por dármelo todo en definitiva y por haberme ayudado a lo largo de este trabajo en cualquier cosa que necesitaba, siendo sujetos de las pruebas hechas, en especial mi padre por haber ido siguiendo muy de cerca todo el desarrollo y ayudándome con la memoria siempre que lo he necesitado. También agradecer a mi madre todo el trabajo que hace por mi día a día que me facilita la vida y me ha permitido finalizar este trabajo con éxito. Los tres han sido un pilar fundamental en mi educación como persona y me han enseñado los valores necesarios para llegar donde estoy.

En segundo lugar, quiero darles las gracias a mis amigos y amigas, por haberme ayudado también en todas las pruebas del proyecto. Agradecerles esos momentos de calma que han sido necesarios para poder conseguir todos mis objetivos con éxito. Siempre me han prestado ayuda y apoyo en todo lo que he necesitado y han contribuido en mis logros, incluido este proyecto.

Por último, pero no menos importante, a todos los lugares donde se me han formado académicamente para llegar a estar donde estoy y haberme ayudado a desarrollar mis capacidades intelectuales y sociales. Al CEIP Caja de Ahorros y el IES Fernando de Rojas donde pase la mayor parte de mi adolescencia, a la Universidad de Salamanca donde he pasado parte de mi juventud y a la Universitat Oberta de Catalunya por haberme ayudado a finalizar mis estudios de la mejor manera posible y especializándome en la rama que siempre quise. Y quiero hacer especial énfasis aquellos profesores y profesoras que me han marcado a lo largo de todo este trayecto y me han ayudado a ser quien soy como estudiante y como persona.

Abstract

Nivel de un juego en el personaje se moverá con teclado o un joystick y la cámara con el ratón o con otro joystick. A la vez se dispondrá de un inventario que almacenará los objetos que recoja por la escena. Estos objetos recogidos se podrán usar con otros interactuables o personajes para poder desbloquear nuevas zonas, diálogos u objetos que estaban ocultos anteriormente. Todo ello llevará a resolver retos propuestos que permitan avanzar en la historia y la aventura. Mezclado con la aventura gráfica se busca introducir zonas de sigilo en las que debemos sobrepasar a enemigos que estén vigilando o incluso acabar con ellos de manera sigilosa.

Referencias del juego pueden ser Broken Sword 3 y 4, El Código Da Vinci o A Plague Tale, en los que se proponen retos intelectuales al jugador mezclado mediante retos o puzles a la vez que hay ciertas zonas de sigilo y más dinámicas.

El objetivo del trabajo es mediante la herramienta Unity y el lenguaje C# crear un nivel totalmente jugable con todas las mecánicas de movimiento, diálogos, inventario y recolección de objetos y puzles como si de una aventura gráfica se tratara. Además de incluir una zona de sigilo como la nombrada anteriormente que se deba recorrer sin ser visto y acabar con algún enemigo en silencio.

Level of a game in which the character will move with the keyboard or a joystick, and the camera with the mouse or another joystick. At the same time, there will be an inventory that will keep the objects that you pick up in the scene. These collected objects can be used with others interactive or characters to unlock new areas, dialogues or objects that were previously hidden. All of this will lead to solve proposed challenges that allow the story and adventure to progress. Mixed with the graphic adventure, the goal is to introduce stealth areas in which we must overcome the enemies that are watching or even kill them silently.

Some references of the game can be Broken Sword 3 and 4, The Da Vinci Code or A Plague Tale, in which intellectual challenges are proposed to the player, mixed with challenges or puzzles, at the same time that there are certain areas of stealth and more dynamic.

The purpose of the thesis is to use the Unity tool and the C# language to create a fully playable level with all the mechanics of movement, dialogue, inventory and object collection, and puzzles, as if it was an adventure game. In addition to including a stealth area like the one mentioned above that must be traversed without being seen and finish off an enemy quietly.

Palabras clave

Videojuego, Aventura gráfica, SCRUM, Unity, C#, GameObject, Prefab, Canvas, asset, NPC, InteractiveObject, Interaction, Item,

Notaciones y Convenciones

Se ha utilizado la cursiva para escribir los *anglicismos*

Se han utilizado las comillas para poner “nombres o textos literales”

Índice

Figuras y tablas

Lista de imágenes, tablas, gráficos, diagramas, etc., numeradas, con títulos y las páginas en las cuales aparecen. Para actualizar cada uno de los índices, hay que hacer botón derecho con el ratón y escoger la opción “Actualizar campos”.

Índice de figuras

Índice de tablas

1.Introducción

1.1. Introducción/Prefacio

Se propone diseñar y desarrollar un nivel completo de un videojuego, el cual debe mostrar todas las mecánicas disponibles a lo largo del juego, a la vez que aspectos importantes dentro de la idea como puede ser la historia y ambientación. La idea es mostrar todos los elementos jugables concentrados en un nivel probado y completamente funcional, para así no excederse en la cantidad y fijarse en la calidad del producto final y que funcione a modo de demo que muestre todas las características que dispondrá el juego final.

Sobre la idea jugable del nivel se basa en esencia en una aventura gráfica adaptada a los tiempos modernos. Lo positivo de este tipo de títulos es que sus mecánicas jugables pueden ser muy versátiles, desde un sistema de inventario y puzles que propongan un reto mental al jugador mediante la mezcla o montaje y utilización de los objetos, hasta puzles que aporten nuevos elementos al título que pueden convertirlo en una experiencia más variada. Además, mediante la narrativa se pretenden introducir zonas que necesiten más de la destreza con los controles del jugador, como lo pueden ser elementos de parkour en las que avanzar suponga un pequeño reto para el jugador, o zonas de sigilo en las que se debe avanzar en silencio sin ser detectado y pudiendo en ocasiones acabar con enemigos que no nos han visto. Pese que las zonas requieran de cierta destreza en muchas ocasiones no dejan de ser puzles en los que el jugador debe tomar una decisión sobre como avanzar para llegar a su objetivo final, aparte de añadir un componente de tensión en momentos que la historia así lo requiera.

Este género está de nuevo en gran auge y con las tecnologías de hoy en día se puede crear una obra capaz de introducir a muchos nuevos jugadores a este tipo de títulos que buscan incentivar una experiencia más narrativa con retos intelectuales. Además de utilizar elementos que juegos muy reconocidos usan hoy en día como lo son el sigilo o los puzles.

La idea nace de un cariño especial desde la infancia a las aventuras gráficas debido que es un género que he jugado durante toda mi vida y me gustaría que mi primer gran proyecto fuera una obra con unas premisas similares, aunque actualizando el concepto. Una de las principales inspiraciones es Broken Sword 3, que pese a que en su lanzamiento no fue muy bien acogido por los seguidores de la saga, tiene elementos que aunque no estén bien desarrollados a priori son interesantes, como introducir la aventura gráfica a un mundo 3D en el que los puzles propuestos pueden disponer de una dimensión más, además de permitirse introducir ciertas secciones enfocadas al sigilo o la aventura pura.

1.2. Descripción/Definición

Como punto de partida se usa la idea plasmada en un pequeño Game Design Document. Por otro lado el desarrollo se apoya en las herramientas de desarrollo, como lo son Unity y su lenguaje C#; y en recursos artísticos como lo son los diferentes *assets* de Unity, entre otros. A partir de este punto de partida se busca crear un nivel totalmente jugable que muestre todas las mecánicas propuestas sobre un juego de aventura gráfica adaptado a la actualidad. La idea es cambiar las mecánicas de movimiento mediante *point & click* por mecánicas de movimiento mediante ejes en escenarios 3D. Este elemento se usa para introducir mecánicas y puzles que permitan explotar este formato como lo pueden ser secciones de sigilo, o una exploración más inmersiva del escenario y una sensación de control más realista por parte del jugador o jugadora.

El objetivo es cubrir esa demanda de juegos más narrativos con mecánicas de acertijos e investigación. Hace años las aventuras gráficas eran el género principal de los juegos para ordenador, pero hoy en día hay un enorme vacío en el género, pero aún tienen mucho público quedando esto patente con el éxito de juego como “Return to Monkey Island”. El principal objetivo es coger el género y “modernizarlo”, aunque las mecánicas más clásicas basadas en el *point & click* siguen funcionando, se busca aunar público más clásico y uno moderno al que los controles con ratón puedan verlo como menos inmersivo o más ortopédico. Todo esto sin olvidar que el juego sigue siendo una aventura narrativa en la que la verdadera importancia está en los acertijos y puzles propuestos, suponiendo la experiencia un reto mental pero ayudado de un lavado de cara de las mecánicas para no alejar público que pueda disfrutar del juego.

El primer paso es diseñar los elementos del videojuego mediante un pequeño *Game Design Document* que describa mecánicas, niveles, historia, personajes y demás elementos relevantes en el juego. A partir del documento se establecen unos tiempos de desarrollo que se deben seguir y unos plazos para cada parte, al tratarse de un desarrollo de una sola persona será imposible trabajar en paralelo por lo que los plazos y tareas serán más fáciles de diseñar, pero se tardan más en desarrollar. Una vez está planificado el desarrollo y se han diseñado las bases en el *GDD* es necesario concretar el nivel a desarrollar en el trabajo, por plazos es imposible desarrollar el juego entero, además de que no es necesario para mostrar las mecánicas y una parte de la historia que nos introduzca, este concepto es muy similar a lo denominado ‘demos’ de los juegos en los que los usuarios pueden probar el juego y ver una introducción a la historia que les deja con ganas de más, a modo de tráiler de película.

Tras concretar el nivel de desarrollo para el trabajo se comienza con el mismo. Lo primero se implementan las mecánicas básicas de personaje como lo son el movimiento y el manejo de la cara, luego se prototipará un nivel y se pulirán las mecánicas básicas ya nombradas.

Una vez se ha realizado esta primera implementación se diseña un boceto en papel del nivel a desarrollar en el que se plasmarán las escalas, arquitectura y demás elementos importantes del mismo, se comenzará a prototipar en Unity. Usando las herramientas de creación de niveles y *assets* disponibles en el motor se implementa una versión inicial del nivel y se colocarán los elementos interactuables, en los que recae gran parte de la jugabilidad y sus mecánicas.

Una vez se tiene un nivel funcional con las mecánicas básicas se comienzan a implementar las mecánicas que necesitarán de los objetos interactuables de la escena. Estas son las mecánicas de coger o utilizar objetos de la escena, hablar con personajes de la escena o usar objetos del inventario de la protagonista con el resto de los objetos de la escena.

1.3. Objetivos generales

Listado y descripción de los objetivos del TF, ordenados por relevancia.

1.3.1. Objetivos principales

Objetivos de la aplicación/producto/servicio:

- Diseñar un nivel de un juego totalmente funcional.
- Diseñar una historia y ambientación.
- Planificar el desarrollo del videojuego
- Proponer retos intelectuales al jugador complejos y divertidos, pero no frustrantes.
- Implementar mecánicas de movimiento de personaje y cámara.
- Implementar mecánicas de objetos interactuables, conversaciones e inventario.
- Comprobar posibles fallos y solucionar para evitar el mayor número de estos.
- Diseñar e implementar los menús e interfaz de usuario del juego

Objetivos para el cliente/usuario:

- Investigar la escena.
- Resolver retos propuestos por el diseñador para avanzar.
- Utilizar las mecánicas para alcanzar los objetivos

Objetivos personales del autor del TF:

- Introducir una historia interesante y atractiva para el jugador o jugadora.
- Diseñar un sistema de mecánicas fluido que no entorpezca la jugabilidad.
- Asociar los retos y enigmas propuestos a la historia dándole un sentido narrativo a cada acción.

1.3.2. Objetivos secundarios

Objetivos adicionales que enriquecen el TF.

- Crear una historia total y nombrar el resto de los niveles que den sentido narrativo al total de la obra pese a solo diseñar e implementar uno de ellos.

1.4. Metodología y proceso de trabajo

1.4.1. Metodología de desarrollo

En este punto es importante tener en cuenta que no hay ninguna metodología obligatoria o consensuada entre los desarrolladores, sino que se elige del tipo de proyecto y de videojuego que se busque crear, el equipo de desarrollo y la financiación, entre otros parámetros. Lo primero a tener en cuenta es que este desarrollo va a ser realizado por una única persona y se busca obtener un resultado final, si bien óptimo, no completo ya que se tratara de un nivel que muestre todas las mecánicas y demás elementos jugables. Sumado a esto, el tiempo de desarrollo es escaso, debiendo diseñar las mecánicas y juego, y a la vez desarrollar un nivel e implementarlo en tres meses.

Por todo lo dicho es recomendable aplicar una metodología ágil que permita dividir el proceso en partes más pequeñas sobre las que ir iterando, y así obtener lo antes posible prototipos jugables para poder testear. Se usará la metodología SCRUM aplicada a videojuegos para el proyecto, facilitando esta elección las tareas ya que es una forma de trabajo ya probada y totalmente funcional. SCRUM es una metodología ideada para cualquier desarrollo software, al aplicarse en este caso a un videojuego hay que adaptar ciertos conceptos que diferencian a éstos de un software normal. Por otro lado SCRUM es una metodología creada para equipos de personas, en este caso al ser un único desarrollador se sustituirán las reuniones, al finalizar cada iteración para comprobar el estado, por comprobaciones del estado y soluciones de posibles fallos encontrados.

El primer paso es dividir la tarea completa en periodos de un tiempo determinado, llamados *sprints*. Tras estos *sprints* se harán una reunión, en el caso de este proyecto una comprobación de errores del desarrollador. Esto será útil para comprobar el correcto funcionamiento del prototipo generado y obtener conclusiones de posibles errores. La solución de los errores debe de añadirse al *backlog* de la siguiente iteración. El backlog es una lista de tareas a realizar, puede referirse a la total del producto o en cada *sprint*.

Inicialmente los *sprints* tendrán una duración superior, esto es debido a que en las fases iniciales no hay muchas cosas a probar que permitan buscar fallos en el producto desarrollado. En el desarrollo de los videojuegos este periodo seria la concepción y diseño de la idea, fase más teórica del proyecto que sentará las bases de todo el desarrollo, pero no dará como resultado ningún prototipo jugable.

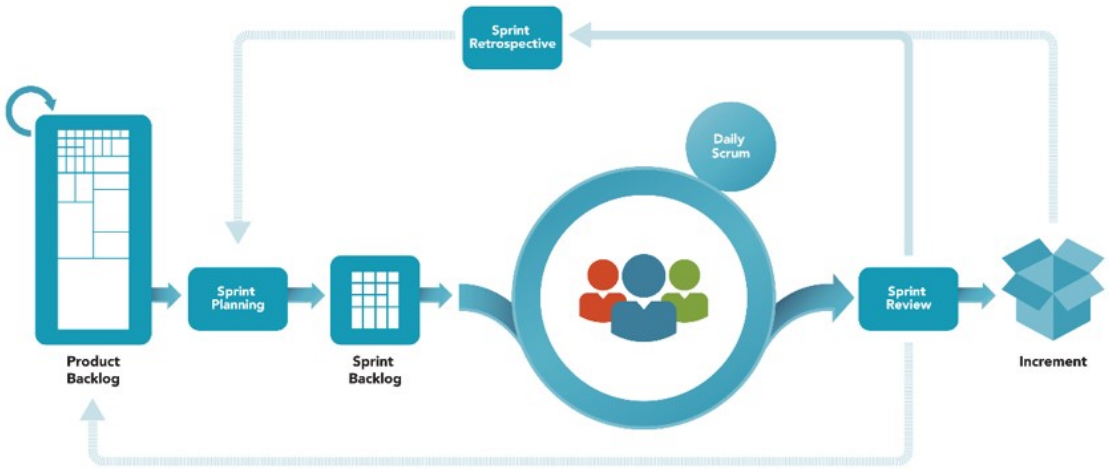

Figura 1: Esquema de metodología Scrum

Como se ve en la anterior figura la idea es obtener todas las tareas necesarias para llegar a una versión final del producto, y a partir de ellas planificar el sprint a realizar. El *sprint* está compuesto por una parte de las tareas, éstas se realizan y tras lo cual se comprueban posibles errores del producto obtenido. A partir de los errores se añaden al *backlog* del producto y por ende serán tareas que realizar en las siguientes iteraciones, para obtener tras cada ciclo de trabajo una versión más completa y con menos errores.

Lo útil de esta metodología es que además de aportar una manera eficiente de trabajar, facilita la planificación de las tareas dándole un sentido total dentro del desarrollo. Las fechas marcadas en la planificación deben de ser la duración del sprint en el que se trate desarrollar la tarea.

1.4.2. Estrategia de trabajo

La estrategia marca como objetivo un producto nuevo. Pese a esto, en lo referente a las mecánicas o algunos aspectos narrativos pueden estar fuertemente inspirados en otras obras audiovisuales o literarias.

El proceso se basa en probar videojuegos similares para poder obtener ciertas inspiraciones, además de detectar ciertos errores de esas obras que se puedan solucionar o mejorar. Hasta día de hoy se han lanzado ciertos juegos que buscan juntar las bases de las aventuras gráficas y actualizarlas mediante los controles y ciertos elementos más actuales de los videojuegos. Uno de los elementos de mayor éxito lo fueron las obras de Telltale Games o de Péndulo Studio.

A partir de aquí se puede crear una narrativa que sea complementaria a la jugabilidad, pero nunca una excusa para que el jugador interactúe con el mundo creado. Es necesario que la historia que se cuente sea interesante y lo suficientemente profunda como para que el jugador empatice y se enganche, a la vez que siente que sus acciones o su ingenio están muy relacionados con todo lo que ocurre a nivel narrativo. Al ser un desarrollo unipersonal no se cuenta con escritores o guionistas que pongan su tiempo y habilidades a esta tarea, por ello para crear una historia es necesario buscar inspiraciones de otras obras que faciliten el trabajo.

Para la creación de niveles, pese que el diseño y la composición de éste será totalmente original, se usarán elementos ya creados por artistas que estén publicados en la web. Crear recursos totalmente originales para el proyecto harían imposible su desarrollo en el tiempo establecido, al ser un único desarrollador, por lo cual la mejor solución es hacer uso de la gran cantidad de recursos que hay en línea.

En conclusión, se trata de una obra totalmente original que trate de aportar nuevos conceptos a un género ya existente, a la vez que cuenta una historia que, sin ser completamente original, sirva para la obra. Además de un mundo diseñado para la obra, pero construido a partir de elementos creados por artistas o diseñadores externos al proyecto.

1.4.3. Metodologías de investigación

Basándose en la metodología de desarrollo se busca obtener prototipos jugables lo antes posibles, estos prototipos facilitarán pruebas en usuarios en fase iniciales. El objetivo es tras iteración SCRUM, el producto jugable que se haya obtenido dárselo a probar a diferentes usuarios para recibir *feedback* de ellos. Toda la información se recolectará por medio de entrevistas realizadas de manera personal a cada usuario que lo haya probado, donde mediante preguntas cortas y concisas se trata de obtener información precisa de posibles errores o mejoras que hayan detectado.

Mediante las iteraciones y las entrevistas realizadas a los usuarios de pruebas se busca ir retocando el producto hasta obtener una versión final completamente funcional, lo más pulida posible y añadiendo mejoras viables no planteadas en el diseño inicial, pero que pueden ser interesantes introducir en un videojuego de este tipo.

Sumado a esto es importante recabar información de otros juegos similares para tratar de usar mecánicas o elementos interesantes. Esta inspiración es muy importante, ya que a día de hoy es muy complicado innovar al completo, pese que hay obras muy originales y diferentes a las demás, sobre todo dentro del ámbito de los *indies*. El resto de los videojuegos suelen ser mejoras de obras anteriores, en las que solucionan fallos o incluyen nuevas mecánicas, pero que en la mayoría de los

casos ya estaban presentes en otros títulos. En definitiva, esta documentación previa es muy útil, y como se ve necesaria en un gran número de proyectos, esto no se traduce en algo negativo, muchas obras a lo largo de la historia de los videojuegos no han sido más que una mezcla de elementos ya vistos, combinados de tal forma que el resultado es mejor a sus propias obras de inspiración. En este sentido se puede ver el videojuego “The Last Of Us” (2013), juego aclamado por crítica y público, que no fue innovador ya que previamente existían juegos con un carácter muy basado en la narrativa, u obras de aventuras en las que se viajaba por el mundo, o de supervivencia con escasos recursos a disposición del jugador; la correcta combinación de todos los elementos y una gran historia convirtió al juego en la obra que es hoy en día.

1.5. Planificación

Debido a tratarse un proyecto orientado a realizar un TFM hay que tener en cuenta que los plazos ya están marcados de una manera general con las entregas de cada fase. El objetivo de la planificación será, apoyándose en la metodología utilizada SCRUM, plantear unas fechas realistas que se enmarquen en las ya propuestas por el trabajo.

Para ello veremos como el desarrollo se divide en las tres fases principales de entrega, para luego sobre ellas dividirlas y tener una planificación detallada que facilite el trabajo y permita ser más preciso en la temporalidad del proyecto. Inicialmente entre el 20 de marzo de 2023 y el 16 de abril se debe desarrollar el estado del arte del videojuego y desarrollar una versión inicial, que sirva como base para a partir de ella comenzar a iterar en el desarrollo y conseguir versiones más perfeccionadas. Tras esto, desde el 17 de abril hasta el 21 de mayo se debe implementar una versión jugable, aquí es donde reside la mayor parte del desarrollo y donde será más importante una buena planificación para poder cumplir con las fechas establecidas. Por último, desde el 22 de mayo hasta el 18 de junio se debe de generar una memoria final de todo el proyecto y un producto habiendo solucionado posibles fallos y añadido todos los elementos necesarios, ya que esta será la versión que se entregue.

Como se ve la primera entrega se dispondrán de 4 semanas, la segunda de 5 semanas y la tercera de 4 semanas, es importante saber del tiempo del que se dispone para así poder planificar las tareas que son necesarias realizar. Para la primera parte es preciso establecer el estado del arte. A partir de las metodologías de investigación establecidas se recaba la información necesaria y se explica en qué estado se encuentran obras similares. Con esta información obtenida se realiza una versión inicial del producto, en la que mediante un prototipo se puedan probar las mecánicas básicas. Por ello se puede utilizar la primera semana para realizar las investigaciones y realizar el estado del arte, una vez se tiene, a partir de ahí, implementar una versión inicial durante 2 semanas, y tras éstas hacer una pequeña prueba con usuarios para ver posibles errores o mejoras, en la última semana previa a la entrega.

Para la segunda entrega la división es más complicada, ya que es donde se concentrará la gran parte del desarrollo del juego. Una vez se comienza a desarrollar es necesario tener claro hasta donde se desea llegar con la implementación y cuales son todos los recursos de los que se dispone, y las limitaciones. Se busca implementar una versión jugable, pero no final, con lo cual se puede continuar prototipando para darle especial énfasis a las mecánicas y jugabilidad. La primera iteración puede durar 1 semana para solucionar todos los errores encontrados e implementar una versión inicial del nivel por la que se moverá el personaje para apoyarse en los elementos de la escena creada y poder implementar el resto de las mecánicas. Tras haber solucionado estos pequeños fallos y tener un nivel jugable, que no final aun, se comienza una iteración de 2 semanas para implementar las mecánicas de inventario, interactuar con objetos de la escena y entablar conversaciones con los personajes.

Además, es importante crear los primeros puzles de objetos que nos permitan mediante una cadena de interacciones de objetos y personajes obtener un objetivo. Una vez realizado esto se debería realizar una prueba con usuario, para en la siguiente iteración añadir a las tareas del *backlog* soluciones a los errores encontrados, de esta manera las 2 últimas semanas se realiza la última iteración dentro de esta entrega, en la que se deberán de solucionar todos los errores para tener una versión pulida y jugable, a la vez que se añaden las mecánicas de sigilo para disponer de ellas implementadas en la entrega de la primera versión jugable, aunque sea en una forma inicial y perfeccionarla de cara a la última entrega.

En la última entrega se disponen de 4 semanas, la primera semana se dedica a detallar todos los pasos del desarrollo en la memoria y a la vez implementar los menús del juego. La segunda semana puede ser otra iteración en la que se realicen pruebas con usuarios y se pula la mecánica de sigilo con toda la información obtenida de las pruebas con usuarios, y a raíz de esto obtener una versión jugable final; en paralelo se continuará desarrollando la memoria. Las últimas 2 semanas antes de la entrega de la versión final se dedican a terminar la memoria de todo el proyecto e implementar un nivel con todos sus *assets* y elementos visuales, este paso será relativamente sencillo puesto que todos los elementos jugables ya están desarrollados. Esta tarea al tener ya realizados prototipos tan solo supone intercambiar las mallas de cubos y esferas por modelos y personajes, y dedicar cierto tiempo a las animaciones. Una vez terminado se realiza una última prueba con usuarios para comprobar el correcto funcionamiento de todo el nivel y que visualmente luce correctamente.

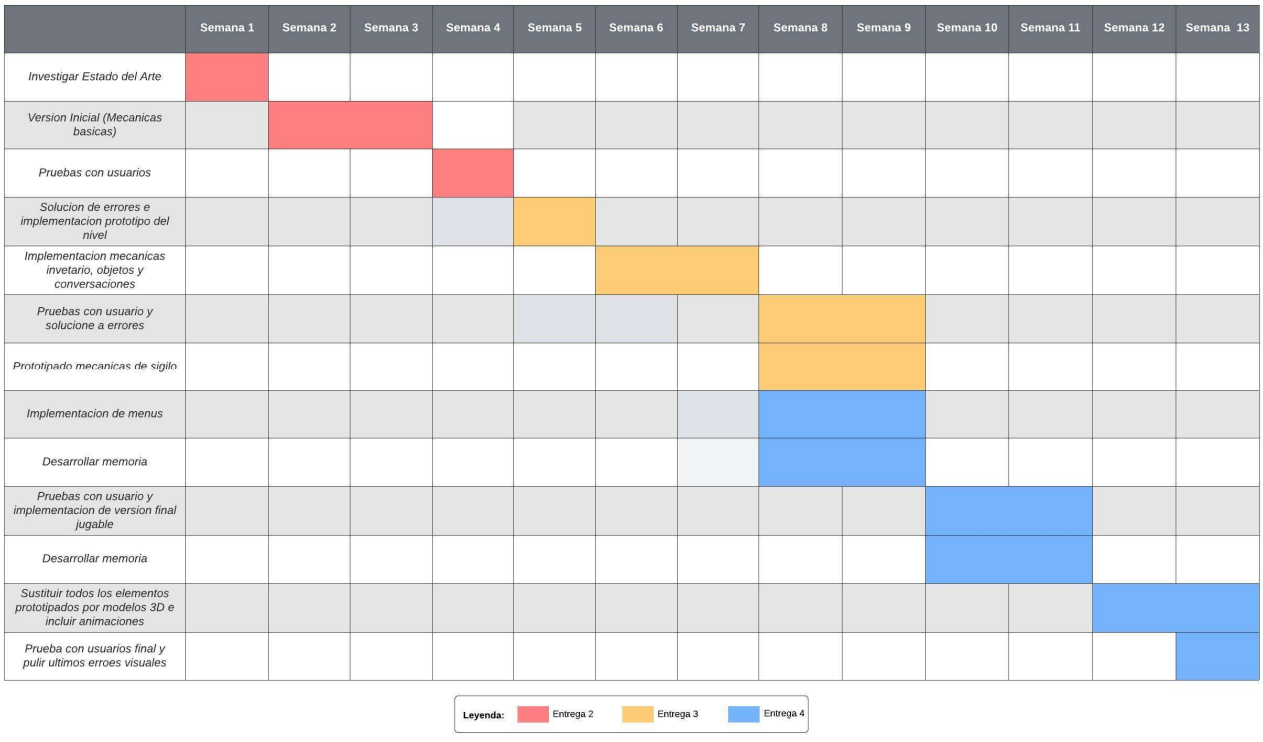

Figura 2: Diagrama Gantt

1.6. Presupuesto

Lo primero a tener en cuenta es que al tratarse de un proyecto orientado a un Trabajo Fin de Master el equipo de desarrollo está formado por un único miembro, facilitando esto el cálculo del presupuesto ya que no es necesario planificar los gastos en sueldos en un equipo de desarrollo. Sumado esto al ser un proyecto dedicado al ámbito educativo se trata de gastar el menor dinero posible.

Los mayores gastos del proyecto serán en forma de tiempo, el cual ya se ha explicado en la planificación. Para calcular un presupuesto aproximado es importante hablar de un tiempo de trabajo semanal, ya que la planificación está hecha por semanas y poder calcular un total de horas aproximadas dedicadas al proyecto. Se calcula que para el proyecto se dedican 20 horas semanales, equivalente a media jornada laboral. Tratándose de 13 semanas de desarrollo serian 260 horas, a las cuales sería necesario sumar la investigación, el diseño y la documentación realizados a parte. Este número de horas se encuentran cercanas a las 300 horas que hay que dedicar a los 12 créditos ECTS que supone un Trabajo Fin de Máster. El sueldo medio en España son 14,60 € por cada hora de trabajo, lo cual supondría un gasto en personal dedicada al desarrollo del videojuego de 3.800 € aproximadamente.

Además del gasto en personal es necesario tener en cuenta el gasto en equipos y recursos necesarios para el desarrollo del software. Lo primero se necesita un ordenador con componentes suficientes para desarrollar un proyecto en 3D. Se calcula que se necesita un equipo de 1000 € aproximadamente, que cuente con una tarjeta gráfica moderna y almacenamiento suficiente para almacenar todos los recursos del juego. Además, sería necesario un CPU y memoria RAM suficiente para que el trabajo sea fluido y el ordenador no suponga un retraso en la planificación del proyecto. Por último, es necesario adquirir algunos *assets* puesto que es proyecto en el que no se cuenta con artistas, modeladores, músicos y demás personal necesario en un videojuego de estas características. El precio de estos recursos es muy relativo y depende de los autores, pero se calcula que aproximadamente pueden ser entre 100 y 500 €. Con todo teniendo en cuenta el peor de los casos el costo de un nivel completamente jugable teniendo en cuenta sueldos y recursos materiales y artísticos puede ser de aproximadamente 5.000€, suponiendo la mayor parte de este en personal, al igual que ocurre en los grandes proyectos en los que están involucrados ingenieros, diseñadores y artistas.

Hay que tener en cuenta que esto es una pequeña parte de un juego, un nivel de un juego, pero también las mecánicas esenciales ya se verían desarrolladas con este pequeño producto. Lo que significa que para finalizar el proyecto es necesario extender el personal para poder perfeccionar al máximo el proyecto y poder generar *assets* propios para el proyecto por diseñadores artísticos y modeladores. Seguramente este incremento en el número de desarrolladores, que como se ha dicho es donde se va la mayor parte del presupuesto de un videojuego, acarrearía un presupuesto

aproximado de 65.000 €. También hay que tener en cuenta que no solo significa un incremento en el personal, si no en la extensión temporal del proyecto para poder obtener un producto final y completo. Gracias al aumento de presupuesto se podría disponer de programadores, artistas, modeladores y diseñadores.

1.7. Estructura del resto del documento

A partir de la introducción se realiza un análisis de mercado en el que se establece un público objetivo en base al tipo de juego que se va a desarrollar, además de establecer posibles perfiles. Esto favorece el correcto desarrollo de las mecánicas y dificultad del videojuego, puesto que no es lo mismo dedicar el juego a un público adulto mayor de 18 años que a niños pequeños. A la vez la historia debe de ir de la mano y mostrar un tono acorde al público objetivo establecido para la obra. También se habla del estado del arte estableciendo en que momento está el mercado y hablando de obras similares para apoyarse en ellas a lo largo del diseño y desarrollo.

Una vez establecido hacia que tipo de usuarios se dirige el juego se establece una propuesta formal del título, estableciendo los objetivos y especificaciones que se van a implementar, a la vez de un modelo de negocio que estará en gran parte relacionado con el videojuego, pudiendo ser un juego como servicio, juego de suscripción, *free-to-play* o compra única, entre otros. En este punto también es necesario hablar de la estrategia de marketing que se va a seguir, elemento muy relacionado con el *target* de la obra y el modelo de negocio, ya establecidos ambos.

A partir de este punto comienza el diseño, es la base del desarrollo, la guía que se establece para seguir durante todas las fases posteriores del proyecto. Es necesario hablar de la arquitectura general, como se almacenará la información del juego; la forma de navegar por los menús y dentro del propio juego; establecer las interfaces gráficas del juego y lo que el jugador ve por pantalla; a la vez que indicar las herramientas de desarrollo y lenguajes de programación que se van a utilizar en la siguiente fase.

La última fase previa a obtener versiones compilables y jugables de la obra es la implementación, en la que se establecen dentro de la memoria los requisitos necesarios de instalación, todos los recursos necesarios. Al mismo tiempo una guía para que el usuario sepa como proceder durante la instalación del juego para tenerlo listo y ejecutarlo.

La penúltima etapa es la demostración en la que se comenzarán a mostrar las instrucciones de uso del juego, como funcionan los menús y controles. Esta demostración está acompañada de los prototipos realizados a lo largo de la implementación y una explicación de los test realizados, y sus respectivas iteraciones y mejoras realizadas siguiendo la metodología SCRUM. Todo puede ir acompañado con

una guía de uso para poder familiarizar a los jugadores con el juego además de ejemplos de como funciona el juego y los diferentes retos, y de qué forma se deben de superar.

El último capítulo es una conclusión en la que se cerrará la memoria estableciendo el producto creado y hablando de todo lo aprendido gracias al desarrollo, se habla de elementos mejorables o elementos que se consideren diferenciadores de otros videojuegos. Por último, se nombrarán las líneas futuras y se establece el camino a seguir para finalizar el desarrollo a la vez que introducir posibles mejoras.

2. Análisis de mercado

Este capítulo pretende hacer un análisis de la situación actual del mercado en el que se enmarca el Trabajo Fin de Máster. Para hacerlo, se estudia el público potencial, su segmentación acompañado de un análisis de la competencia y la situación actual del mercado en el ámbito de los videojuegos tratando de buscar las fortalezas de la obra e intentar paliar al máximo sus debilidades.

2.1. Público objetivo (i.e. target audience) y perfiles de usuario

La audiencia objetiva de un videojuego puede ser muy diversa, pese a que se establezca un perfil de usuario que se considere que va a jugar el producto, no significa que sea excluyente para otro tipo de usuarios. En este punto es muy importante destacar la selección de dificultades que tienen la mayoría de los juegos, esto permite en función del tipo de jugador elegir una experiencia u otra. En caso del videojuego a desarrollar al proponer retos intelectuales al jugador es complicado regular una dificultad, sumado a que el usuario puede no tener claro que dificultad es la ideal para él. Debido a esto es aconsejable incluir pistas a modo de ayuda a los usuarios para que las usen cuando las necesiten en cada situación, permitiendo esto ampliar el *target* del videojuego.

Pese a intentar abarcar un público más extenso se crea un perfil de usuario objetivo para conocer los gustos y necesidades de los jugadores que se buscan satisfacer. Lo primero al hablar de perfiles de usuario es ir de lo general a lo concreto porque hoy en día existe mucha oferta de videojuegos y de muy diferentes tipos. Entre ellos hay desde jugadores más casuales que juegan diferentes títulos en sus móviles, con una baja frecuencia, hasta jugadores que dedican muchas horas a juegos online competitivos, o amantes de grandes retos propuestos por los desarrolladores.

Según DevUego existen tres grandes grupos de jugadores, los *casual gamers*, los *mid gamers* y los *hardcore gamers*. El juego busca ir orientado a los dos primeros, en los que el control sea relativamente sencillo, sin muchas acciones y que los retos residan más en el ingenio, aunque a su vez se trata de contar una historia profunda con unos valores de producción más altos, de lo que pueden ser los juegos móviles. La idea es que el juego no busque consumir muchas horas de los jugadores, sino que lo puedan ver a modo de una serie, en la que echándole un rato al día irán avanzando en la aventura. La búsqueda del público objetivo del juego es muy útil a la hora de diseñar los desafíos que se proponen al jugador y su dificultad, ya que es necesario encontrar un equilibrio en el reto propuesto sin que caiga en ser aburrido, pero tampoco que llegue a ser frustrante para los jugadores. Aquí es donde reside una tarea muy importante de los diseñadores del juego y de los niveles, en función del tipo de juego. En este caso se proponen tanto retos mentales como retos de habilidad, como el de

sigilo, en ambos casos es necesario ajustar la dificultad para obtener un producto más optimo en función del público objetivo.

Tras haber definido el *target* el desarrollo se favorece de la metodología de trabajo SCRUM, ya que en las diferentes pruebas con usuarios que hay en las iteraciones se puede testear con usuarios de diferentes perfiles para ver como reaccionan hacia el juego, y si la dificultad de la jugabilidad y retos propuestos es la adecuada o es necesario modificarla.

2.2. Competencia/Antecedentes (o marco teórico/estado del arte)

Como ya se ha explicado el juego no es una obra completamente original, ya que hoy en día es prácticamente imposible diseñar un videojuego totalmente novedoso, sino que más bien son reinvenciones y ensamblar elementos jugables ya existentes. Por eso resulta muy positivo para dar inicio a la primera iteración recabar información de obras similares o usadas como inspiración. Esta información sirve para ver lo que funcionó en aquellos videojuegos y lo que no funcionó de cara al público al que nos dirigimos, que será similar al que esos juegos iban dirigidos.

Se habla en profundidad de varios títulos, en orden cronológico, en los que se ha inspirado y se usan como antecedente para obtener cierta información sin necesidad de probar conceptos en pruebas con usuario y obtener *feedback*. Los títulos de los que se va a hablar comparten elementos con la idea del videojuego además de ser muy importantes dentro del género.

El primero es “The Secret of Monkey Island”, este juego es importante no porque a nivel mecánico sea muy similar sino porque fue uno de los grandes precursores de las aventuras gráficas. Monkey Island establecía un tono de humor en el que al personaje le ocurren una serie de desgracias que se interponen en su objetivo de ser el pirata más famoso de todo el Caribe, para superar estas desgracias los jugadores tienen que hacer uso de su ingenio para interactuar con los objetos del escenario fijos y con los que se puedan recoger, y hallar soluciones a los rompecabezas y puzles a lo largo de la aventura. Pese a que Monkey Island estuviera escrito en un tono humorístico muchas obras que compartían jugabilidad fueron más serias y también gozaron del mismo éxito, como lo pudieron ser “Doom” o “Indiana Jones and the Fate of Atlantis”. Aunque a nivel jugable Monkey Island es muy diferente a la obra que se va a desarrollar, muchas ideas de acertijos o mecánicas como la batalla de insultos pueden ser muy interesantes a tener en cuenta. Más allá de las limitaciones de la época, en las que también reside parte del encanto de la obra, Monkey Island fue una obra redonda, donde comenzó a darse a conocer el género, y aclamada por crítica y público. Seguir los conceptos que se sentaron como base en aquel videojuego es un completo acierto, y a lo largo del diseño servirá como base para pensar en los retos mentales que se proponen a los jugadores.

El siguiente juego, el cual es el antecedente más similar, es “Broken Sword 3: The Sleeping Dragon”, este juego no solo comparte el género de las aventuras gráficas, sino que a nivel mecánico se acerca bastante a la idea propuesta anteriormente. La historia habla de las aventuras de George Stobbart y Nicole Collard los cuales a lo largo de toda la saga investigan antiguas leyendas que a la vez estaban compuestas por algunas sectas con fines mucho más perversos. En esta tercera entrega la jugabilidad cambió por completo de ser juegos *point & click* a un juego con unos controles más modernos y adaptados a las consolas y sus mandos de la época. La siguiente entrega la saga mantuvo el control en un eje 2D con teclas o joysitck, pero volvió a introducir el movimiento con el ratón por las críticas que recibió Broken Sword 3. Este videojuego resulta muy útil como antecedente, no solo por la gran similitud y las posibles inspiraciones que pueda aportar, sino porque fue un juego que tuvo muchos fallos y no fue del agrado de la mayoría de los fans. Este espacio de mejora para los jugadores del género de las aventuras gráficas es el que se debe aprovechar, mejorando los posibles fallos que pudiera contener la obra, pero manteniendo ciertas ideas útiles para explotar un género como las aventuras gráficas en las tres dimensiones.

De este juego las conclusiones que se sacan es que muchos amantes de las aventuras gráficas buscaban en ellas que les contaran una buena historia en la que mediante unos controles sencillos y agiles se pudiera resolver los acertijos y puzles propuestos, Broken Sword 3 peca en esto, ya que al introducir el 3D como un elemento novedoso de la época incluyó todos los fallos que esto acarreó, sumado a un abuso de puzles que hicieran uso de él. El problema es que muchos puzles que hacían uso del mundo 3D eran lentos en muchas ocasiones, repetitivos ya que siempre era la misma mecánica y se recurre constantemente de ellos dejando de lado otros aspectos mucho más importantes en las aventuras gráficas: como lo son los retos intelectuales de combinación de objetos, las conversaciones e interacciones con los personajes, los puzles que se proponen durante la aventura y otros. Además, Broken Sword 3 tiene un problema de ritmo, tiene un comienzo muy dinámico y presenta una historia innovadora (o comienza de una forma muy dinámica presentando una historia innovadora), pero pronto cae introduciéndonos en niveles lentos o en los que los acertijos no tienen ningún sentido, haciendo que el jugador se aburra y esto repercute de manera directa en la historia, ya que ésta se usa como motor de avance durante la aventura. En definitiva, todos estos aspectos son necesarios tenerlos en cuenta a la hora de diseñar el juego para evitar caer en los mismos errores que obras similares anteriores.

Otras obras parecidas de aquella época que sirven como antecedentes pueden ser “El Código Da Vinci”, basado en la novela homónima, que era una aventura gráfica en 3D con zonas de sigilo; o “Tomb Raider 6: Ángel de la Oscuridad”, que pese a estar orientada más a la acción contaba con zonas de sigilo y parkour, y momentos en los que la protagonista debe hablar con otros personajes o usar objetos del inventario para avanzar. Todos los juegos similares de aquella época tenían los mismos problemas con los controles, siendo demasiado toscos. Esto es uno de los principales problemas a

tener en cuenta al diseñar e implementar las mecánicas de movimiento, ya que los jugadores de este tipo de juegos quieren controles ágiles que les lleven rápidamente a donde quieren, no teniendo que perder tiempo en animaciones inútiles o movimientos que no aportan nada.

Por último, un antecedente más reciente, que pese a no compartir el género cuenta con un elemento muy interesante dentro de este tipo de juegos, es la saga de “A Plague Tale” compuesta por dos videojuegos. En ambos títulos la protagonista atraviesa zonas de sigilo en las que su habilidad física no servirá de nada, puesto que al ser más inferior a las de sus enemigos, tendrá que hacer uso de su ingenio y de los elementos de los que dispone a su alcance para sobrepasarlas. Además, estas obras cuentan con niveles que son en sí mismo puzles que explotan la faceta del mundo 3D, dándole un sentido por sí mismo al nivel. Esto resulta muy atractivo para los jugadores puesto que la necesidad de moverse para ir resolviendo un puzle es también la que le impulsa a continuar.

Figura 3: Captura de un puzle de “A Plague Tale”

Resulta interesante parar a observar ciertos puzles de este juego, como el visto en la figura 3, en el cual mediante el movimiento de unas antorchas debe abrirse camino hasta la puerta de un castillo. El jugador nunca podrá moverse por zonas oscuras del escenario, por las que nunca haya pasado la luz, ya que hay ratas que le devoran. Este puzle usa el propio escenario como plano para generar un puzle en 3D en el que el jugador debe usar no solo su ingenio sino su inteligencia espacial para resolverlo. Este tipo de puzles enriquecen el diseño de niveles y pueden ser útiles para darle un sentido a bloquear ciertas zonas y la única manera de avanzar es resolverlo, siendo esto más profundo que una puerta cerrada y un panel numérico en el que introducir una contraseña que se deba buscar.

	El legado olvidado	The Secret of Monkey Island	Broken Sword 3: The Sleeping Dragon	A Plague Tale
Género	Aventura Gráfica	Aventura Gráfica	Aventura Gráfica	Aventura, sigilo
Edad recomendada	+12	+12	+12	+18
Ambientación	Actualidad	Época pirata	Actualidad	Época medieval
Movimiento personaje	Eje 2D	Point & Click	Eje 2D	Eje 2D
Movimiento cámara	Alrededor del personaje	Sigue al personaje	Sigue al personaje	Alrededor del personaje
Mecánica principal	Inventario y puzles	Inventario y conversaciones	Inventario y puzles	Sigilo y puzzles
Importancia a la historia	Si	Si	Si	Si

Tabla 1: Comparativa entre otros títulos del mercado

2.3. Análisis DAFO

	Interno	Externo
Negativo	Debilidades: - Assets no originales debido a no disponer de presupuesto y tiempo para crearlos específicamente para el juego - El tiempo de desarrollo y el numero de desarrolladores impiden crear e implementar todos los niveles de la historia, la cual es un punto fuerte de este tipo de juegos	Amenazas: - Momento en el que están saliendo muchas aventuras gráficas nuevas (aventuras narrativas) o continuaciones de sagas antiguas (“Return to Monkey Island”), esto puede provocar comparaciones y mayor criterio por parte de los jugadores. - Debido a ser un proyecto enfocado a un trabajo fin de master, y poder ser catalogado como indie en base a los recursos disponibles es imposible competir con empresas a nivel marketing y publicidad lo cual en ocasiones provoca que la calidad no vaya de la mano del éxito. -
Positivo	Fortalezas: - Proyecto económico gracias a usar Unity (motor gráfico gratuito) y *assets* de uso libre en internet.	Oportunidades: - Gracias a desarrollar en Unity podemos exportar el juego a distintas plataformas con una mayor facilidad - Por el género elegido posibilidad de expansión y nuevas mecánicas mediante puzles que tengan sentido dentro de la historia - Al ser un género antiguo posibilidad de innovar con el e introducir elementos de otros géneros que mejores la experiencia y la actualicen.

Tabla 2: Análisis DAFO

3. Propuesta

A partir del análisis de mercado hecho en el capítulo anterior, este tercer capítulo pretende explicar de manera resumida la propuesta de la aventura que se pretende crear. Se comenzará explicando los objetivos del videojuego, así como las especificaciones que permiten cumplir estos objetivos y se harán especial énfasis en las particularidades que lo diferencian de la competencia, las cuales se han explicado ya en el anterior capitulo para ver las posibles fortalezas de la obra.

3.1. Game Design Document

En este punto al estar hablando de los videojuegos es interesante explicar el GDD. Este documento es el camino a seguir que se marca desde la concepción de una idea. Muchos de estos documentos son escritos en sucio inicialmente y luego se van puliendo para que adopten una estructura más formal. En nuestro caso el GDD se ha ido escribiendo a lo largo de todos los capítulos anteriores en paralelo, mientras se introdujo los objetivos, se analizó el mercado y los posibles puntos positivos y negativos del juego y sobre todo en la concepción de la propuesta. Este documento en ocasiones es reiterativo con lo dicho en la memoria, pero es interesante tener una sección dedicada a resumir todos los conceptos clave del videojuego, ya que este capítulo es el último previo a la implementación y en este punto es ya de vital importancia para el desarrollo conocer lo que se quiere hacer y lo que no, teniendo una guía que seguir.

3.1.1. Conceptos generales

Nombre del juego: El Legado Olvidado / The Forgotten Legacy

Género: Aventura gráfica en 3ª persona

Mecánicas básicas:

- Moverse por la escena
- Recoger objetos
- Mirar objetos
- Interactuar con objetos
- Hablar con personajes de la escena
- Usar objetos del inventario con otros objetos del inventario o de la escena o con personajes
- Resolver puzles

Modo de juego: Un jugador

Plataforma: PC

Lenguaje de programación y motor usado: C# en Unity

3.1.2. Mecánicas y lógica de gameplay

El jugador o jugadora mueve a un personaje por un mapa en el que debe resolver un enigma o misterio que le permita avanzar en la historia. Para resolver estos puzles es necesario hablar con los otros personajes para conseguir información a la par que recoger objetos del escenario que se almacenan en el inventario. Los objetos se pueden combinar con otros para obtener soluciones a retos propuestos o dárselos a personajes para que nos ayuden a continuar.

Un ejemplo de un puzle de este tipo seria que el jugador deba avanzar por una puerta cerrada con llave, la cual la tiene un guardia de la escena. El guardia no nos va a dar la llave, pero dice que tiene sed y que si le podemos dar algo de beber. Para resolver esto deberíamos de conseguir una bebida y echar algo dentro de ella para que al beberla el guardia le deje fuera de combate y podamos robarle la llave. Esto sería un ejemplo básico, pero con las mecánicas explicadas se puede ampliar y complicar todo lo que se desee.

La idea de estos retos es que se propongan al jugador del final hacia el principio, el jugador sabe que necesita conseguir algo, y para conseguir ese algo deberá realizar otra serie de tareas, y esa serie de tareas también necesitan otras a su vez. Esto se puede complicar todo lo que se desee y obliga al jugador a usar la inventiva y la deducción para que, sabiendo lo que necesita, poder descubrir que pasos debe tomar. Este proceso se basa en la jugabilidad como reto mental para el jugador.

Además de los propios retos propuestos mediante el sistema de mecánicas de objetos del inventario también existirán puzles a lo largo del juego. Un ejemplo de esto pueden ser los típicos minijuegos de móviles que mediante la destreza mental el jugador debe llegar a un estado del puzle que le permita avanzar.

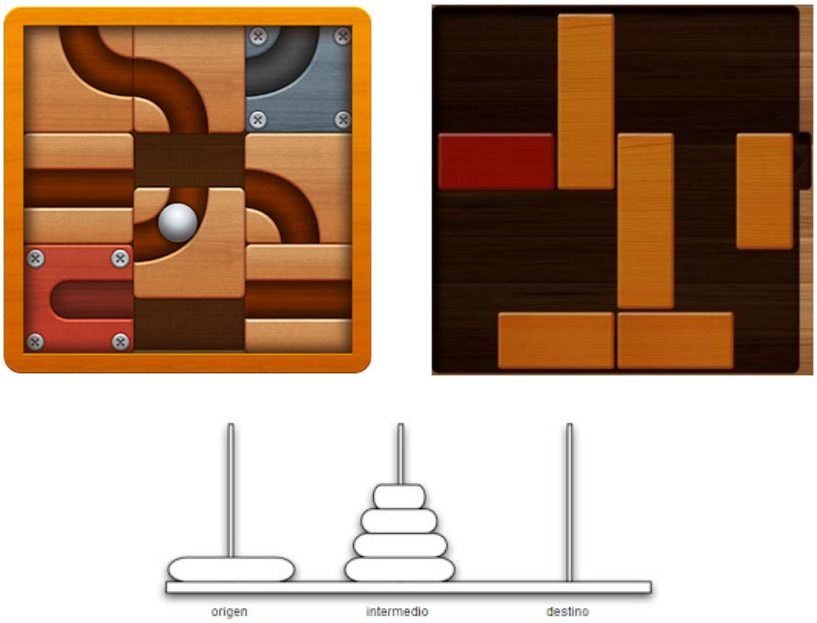

Figura 4: Puzles típicos a incluir en aventuras gráficas

De nuevo estos puzles se pueden complicar todo lo que se desee añadiendo nuevas mecánicas, o incluso siendo necesaria información del propio juego para resolver algunos puzles.

Además de los retos mentales ya explicados el juego contará con zonas en las que el jugador debe esquivar algunos peligros de forma sigilosa, haciendo uso de sus habilidades físicas. Estas secciones serán completamente en sigilo permitiendo en ocasiones al jugador de esta forma acabar con algún enemigo. El jugador debe avanzar sin ser detectado, nunca se buscará un enfrentamiento directo mediante armas, sino que siempre la idea es añadir tensión a la jugabilidad mediante secciones más enfocadas en la destreza física de los jugadores.

3.1.3. Historia y elementos narrativos

- Sinopsis historia:

Una profesora de historia de la Universidad de Salamanca (España) le llega una misteriosa carta, de una persona anónima, en la que le explican que necesitan hablar con urgencia y que quiere entregarle un objeto perteneciente a su familia que se robó hace cientos de años. La protagonista acude al lugar y una vez allí descubre que la policía ha cerrado el edificio y se ha cometido un asesinato. Aquí comienza una investigación que le llevará a descubrir la entrada a unas catacumbas secretas bajo la ciudad de Salamanca y una leyenda que hoy en día sigue más vigente de lo que parece. Para destramar los misterios la protagonista deberá averiguar qué tienen que ver ella y su familia. La aventura presentará una serie de retos y acertijos a través de diferentes lugares de España y Portugal para descubrir que

todo está relacionado con uno de los eventos más importantes de la historia de España, el descubrimiento de América.

• Historia completa:

A modo aclaratorio la historia está inspirada en hechos que ocurrieron de verdad y leyendas de la ciudad de Salamanca. A partir de estas leyendas se han ficcionado los hechos para contar un relato más interesante que se aleja completamente de la realidad

La historia se basa en las leyendas de Salamanca del siglo XV. El primer factor a tener en cuenta es que Salamanca está construida sobre una red de túneles laberínticos en los cuales las leyendas cuentan que han ocurrido reuniones de sectas y experimentos inhumanos. Además, en el siglo XV hubo un enfrentamiento entre dos bandos en la ciudad de Salamanca en la que tras una riña por una pelota dos hermanos mataron a los hijos de María Rodríguez de Monroy, cometido tal crimen huyeron a Portugal por miedo a las represalias.

A lo largo de la aventura el jugador descubre que Cristóbal Colón en el tiempo que pasó en Salamanca (en el convento de San Esteban) tuvo acceso a las catacumbas de Salamanca. Allí descubrió que una secta se reunía en secreto en las catacumbas y cuando quiso contárselo a los Dominicos y destapar todo le amenazaron, ya que la secta tenía mucho poder de influencia sobre los poderosos de la ciudad de Salamanca, y revelarlo podría suponer perder su posible financiación para el viaje que quisiera emprender. Colón decidió no revelar la información sobre la secta, pero dejó para sus descendientes la información para que continuaran la investigación y así poder descubrir si era algo más grande.

La protagonista, llamada Sara, al comenzar la aventura descubre que ha sido llamado por su tío segundo, el cual era descendiente de Cristóbal Colón y había continuado la mencionada misión. El problema es que la secta descubrió estas investigaciones y buscaba acabar con su vida. Nada más empezar la protagonista descubre que ha fallecido y el parentesco que tiene con él, además esta investigación le lleva a una entrada de las cuevas de Salamanca, lugar que Colón usó como estudio para realizar sus estudios e investigaciones sobre los viajes. Allí Sara descubre como Colón decidió no contar nada pero que guardaría la información en su próximo destino, ya que temía que en Salamanca pudiera ser descubierta. Esto lleva al protagonista en una búsqueda por lugares emblemáticos por los que viajo Cristóbal Colón, estableciendo su próximo destino en Córdoba, lugar donde Colón estuvo tras su paso en Salamanca.

Tras estas investigaciones la protagonista descubre como Colón investigó que en la ciudad de Salamanca 20 años antes de que el llegara hubo unas trifulcas familiares, en las que María Rodríguez de Monroy (apodada María La Brava) decapitó en Portugal a dos hermanos que habían asesinado a su hijo por una trifulca. Esto llevará a sara hasta Portugal para descubrir que es lo que ocurrió y como se

formó años después una secta que operaba en las catacumbas de Salamanca. En Portugal Sara descubre que uno de los bandos fue el que formó la secta 10 años después, en 1476, tras firmar un acuerdo de paz que no consideraron acorde a sus pretensiones. En Portugal la protagonista descubre que Colón volvió a Portugal tras su paso por Salamanca, comprendiendo así que él antes de partir en su histórica travesía continuó investigando los hechos acontecidos en Salamanca y descubriendo que los resultados de sus descubrimientos están en Palos de la Frontera (Huelva), ciudad del puerto desde donde partió.

Tras viajar a Palos Sara busca el último lugar donde residió Colón antes de partir en su primer viaje, buscando algún documento que le permita descubrir toda la verdad. En un diario encontrado, en el cual deja en legado a sus herederos continuar la investigación, se revela que volvió a Salamanca en secreto donde estuvo unos meses investigando los hechos y todo le llevo a María La Brava, a la cual persuadió para finalmente descubrir que ella no tenía nada que ver con la secta que ~~él~~ había descubierto. Durante estos meses mantuvo un pequeño romance con María, la cual quedó embarazada de él antes de que partiera de nuevo a Palos de la Frontera para emprender su travesía. Tras esto Sara descubre que no es solo descendiente de Cristóbal Colón, sino también de María y este hecho es el que provoca que la secta del bando rival a María haya perseguido a todos sus descendientes, puesto que compartían sangre con la que hace seis siglos juraron venganza. Para finalizar la protagonista volverá a Salamanca y denunciará los hechos descubiertos para detener a la secta y así evitar que sigan cometiendo todo el mal que ya han causado durante esos últimos siglos persiguiendo a todos sus antecesores.

- Niveles del juego según la historia:

Apartamento del tío segundo de la protagonista
Entrada a las catacumbas de Salamanca
Catacumbas de Salamanca y estudio de Cristóbal Colón
Córdoba
Portugal
Palos de la Frontera, Huelva

3.1.4. Puntos de interés

- Controles:

El jugador se controlará mediante dos ejes (dos joysticks, o 4 teclas de un teclado y el ratón). El primer eje moverá al jugador por la escena siempre pegado al suelo. El otro eje moverá la cámara por la escena para permitir al jugador buscar puntos de interés a su alrededor.

Por otro lado, se permite un botón de interacción el cual en función del punto de interés cercado realizara una acción u otra. Si es un objeto que se puede recoger se guardara en el inventario, si es un

objeto que mirar el personaje dirá una frase ofreciendo cierta información, si es otro personaje se iniciara una conversación.

• Que lo hace divertido:

El objetivo del juego es proponer retos mentales a los jugadores que a través de la lógica y la deducción busquen soluciones a los problemas que proponga el juego. Hay que buscar balancear estos retos, ya que deben de ser lo suficientemente complejos y desafiantes como para que el jugador no se aburra, pero nunca llegando al extremo de que carezcan de sentido o sean demasiado complejos para evitar que el jugador abandone el juego por frustración.

Para añadir variedad al juego se introducen ciertos minijuegos que dinamicen la experiencia y propongan nuevos retos más allá de las mecánicas de inventario y conversacionales ya explicadas. Estos minijuegos suponen cierto aire fresco y una forma de que el jugador pueda desconectar de lo que son las mecánicas principales, para evitar que se agobie, y así se distraiga con estos puzles.

Como elemento clave, en juegos que buscan proponer retos mentales, se deben introducir mecanismos que atrapen al jugador dentro de la experiencia y le generen conocer o descubrir que hay más allá. En algunos juegos a través del arte se pueden buscar experiencias únicas para los jugadores, en este caso se busca que mediante la narrativa el jugador se enganche a una historia de la que quiera saber más, y eso genere en él un incentivo para resolver los retos propuestos, siempre sin caer en los errores comentados anteriormente. Es necesaria una historia que atraiga al jugador porque quiera seguir descubriendo, no es necesario que sea emocionalmente compleja, aunque puede serlo, pero el objetivo principal de la narrativa debe ser contar una autentica aventura, una experiencia que haga que el jugador se identifique con la protagonista y que se apropie de esa necesidad que el personaje principal tiene de conocer más sobre la historia mientras viaja por diferentes zonas del mundo. Es interesante ver como en estos tipos de juegos, aventuras gráficas, los protagonistas por norma general son personajes normales, que carecen de cualquier habilidad espectacular, en contraposición a muchos juegos de acción, aquí se buscan personajes más simples para que el jugador empatice con él y sienta suya la propia aventura (“Monkey Island”, “Broken Sword”, “Runway”, entre otros son ejemplos de protagonistas que podrían ser denominados “pringaos”).

Por último, se añadirán ciertas zonas de sigilo en las que se busca que el jugador desconecte totalmente de retos mentales, y pese a que en ocasiones puede ver los patrones de enemigos o en qué punto es conveniente atacarlos en silencio para poder continuar sin ser visto, estas zonas aportarán un extra de tensión cuando la historia lo requiera, sin abusar de ellas para no perder esa emoción que se busca crear en el jugador, y con esa sensación de peligrosidad.

3.1.5. Tipos de retos y puzles

En las aventuras gráficas hay retos y puzles. Los retos son aquellos que proponen un problema al jugador el cual tienen que solucionar mediante los objetos y personajes con los que puede interactuar; los puzles es introducir acertijos ya creados o inventarse alguno nuevo para resolver un problema concreto, por ejemplo, para abrir una puerta el típico puzle de forzar la cerradura como en “Skyrim”.

Para conocer hasta donde se va a poder llegar se van a dividir los retos que se vayan a proponer según el tipo de objetos que interactúen en él y según su estructura. Para la división por tipos se hará una lista general de todos los tipos de retos que se pueden proponer en una aventura gráfica según R. Fervenza:

• Combinación

Una de las categorías mas importante dentro de las aventuras gráficas, incluyendo la unión entre ítems del inventario o usar estos con otros objetos y personajes de la escena. Es la base de cualquier reto propuesto en este tipo de juegos.

• Combinar dentro Inventario

El jugador o jugadora debe juntar dos elementos del inventario, los ítems formando uno nuevo o ejecutando alguna funcionalidad necesaria para continuar con la aventura. Este punto debe suponer un reto y no resultar evidente, pero tampoco demasiado enrevesada, para poder balancear la dificultad del juego. En ocasiones podremos generar retos más arbitrarios de combinación, pero deben de tener alguna justificación narrativa previa. Por ejemplo, en los juegos de Monkey Island se establece que los pollos de goma son objetos útiles que pueden servir para atravesar tirolinas, esto es debido a que previamente hablando con los personajes de la aventura tiene una justificación lógica dentro del mundo ficticio creado por Ron Gilbert.

• Entorno

En este caso en los retos intervendrán un item del inventario y un objeto de la escena en la que esta el jugador. No todos los objetos del mundo se pueden coger y añadir al bolsillo, por razones obvias, y alguno de estos objetos permanecen a la espera de que se use sobre el un item que combine con el correctamente. En ocasiones las escenas están habitadas por personajes, los cuales pueden ser que no nos permitan avanzar a no ser que le entreguemos algo, en cuyo caso se debe buscar la manera de conseguir ese item y entregárselo al personaje.

Los objetos de la escena deben destacar entre los demás de algun modo para que se pueda diferenciar cuales son mera decoración y cuales tienen una funcionalidad asociada. Aunque tambien es cierto que tampoco debe ser excesivamente evidente para añadir una complejidad mayor a los

retos, pero lo aconsejable es que se permita algun tipo de pista que los remarque en caso de que el jugador o jugadora se quede atascado.

De nuevo se aplica el mismo concepto de dificultad que en el anterior tipo, se debe de seguir una lógica básica para generar estos puzles, sino se terminara por aburrir al jugador. Aunque siempre hay libertad para implementar puzles con lógica dentro del mundo creado.

Ejemplos: Volvamos a la colmena que cuelga del árbol. Queda fuera de nuestro alcance. Está demasiado alta y nuestro personaje no es capaz de subir al árbol. Cerca de allí juegan un par de niños con unos palos. Subimos a uno de ellos para que golpee la colmena y la desprenda, por supuesto no pondrá reparos, con tal de cargarse algo…

- Recetas

Este tipo de retos es de los más conocido, y es muy parecido a la combinación de ítems. En esta caso la diferencia es que los pasos habituales es que el jugador obtenga de alguna forma una lista de ingredientes que deberá de buscar por el escenario, para posteriormente juntarlos y obtener un item compuesto de todos los demás. Puede resultar muy útil para crear un nivel en base a el poniéndose como objetivo generar este mega-item que, de alguna forma explicado mediante la narrativa, permite finalizar esta parte de la aventura.

Una practica muy utilizada en este tipo de retos es no darle todo el trabajo hecho al jugador, sino que alguno de los ingredientes deba ser sustituido por otro que se le asemeje. Esto puede introducirse en de una manera humorística, pero tampoco se debe intentar sustituir ese ingrediente por algo que no tenga sentido o se podría desbalancear la dificultad.

Ejemplo: Damos con la cabaña del guarda forestal. Revolviendo en sus cosas encontramos una relación de plantas que mezcladas funcionan como somnífero para osos. Que si escalar a lugares de difícil acceso para encontrar alguna de ellas, que si remplazar alguna con una hierba similar por estar fuera de temporada, que si pelear con un animal por otra…

- Equipamiento

La clave de este tipo de puzle reside en permitir al jugador o jugadora usar los objetos del inventario. La gran mayoría de ítems no tendrán una funcionalidad asociada y solo reproducirán un dialogo que sirva de pista para cual es el uso del item. Pero habrá ciertos ítems que cuando se interactue con ellos permita a la protagonista equipárselo y avanzar por una zona, que sin el equipo le hubiera sido imposible.

La forma ideal de utilizar este reto es combinándolo con una receta, para que el jugador de forma natural sepa que no puede pasar cierta zona hasta obtener un item complejo, pero comprenda que puede crearse a base de otros que tiene o ha visto por la escena.

Ejemplo: Comprobamos que aunque al oso le encantan la miel y los hombres, no tiene ningún interés en las ovejas. Con algunos objetos de nuestro inventario fabricamos un disfraz ovino (equipamiento) para poder pasar por delante de él sin que nos moleste.

• Intercambio

Anteriormente se habló de usar ítems con objetos de la escena, y que esto incluía los personajes. Habrá situaciones en la que la motivación para entregar un objeto a un personaje de la escena no es que se ejecute alguna acción que permita avanzar, sino que este personaje nos entregue algo a cambio que forme parte del resto de puzles necesarios para finalizar el nivel.

Es importante que el *item* que forma del intercambio no este dicho explícitamente dentro de la narrativa, sino que se genere la idea al jugador de que un personaje necesita algo en concreto en base a la información que obtenga mirándolo y hablando con el. En este caso los diálogos cobran especial importancia para dar información necesaria para saber que necesita el personaje. Este puzle también requiere que el jugador sea consciente de que el personaje le va a devolver algo que el necesita, si generamos puzles muy enrevesados se puede hacer creer al jugador que no necesita lo que el personaje le va dar y complicar todo en exceso.

Ejemplo: Hay un puesto ambulante de venta de miel. La dependienta debe de tener una sangre muy apetitosa porque los mosquitos no dejan de picarle en la cara. Nuestro personaje, que no tiene nada en los bolsillos (¿qué personaje de aventuras lleva dinero encima?), necesita un poco de miel para deshacerse del oso. Con unos objetos de nuestro inventario, una red, por ejemplo, improvisamos una mosquitera que le cambiaremos por un tarro de miel.

• Sucesiones

Para estos retos suele intervenir una sucesión de retos más simples, son muy importante para poder generar un nivel completo de una aventura.

- Sucesos

En ocasiones para presentar ciertos retos o dar la posibilidad de que se resuelvan los ya propuestos es necesario que suceda un evento previo. Esto es lo que hace que no todo se pueda hacer a la vez o que no de igual el orden en el que se realicen las cosas. No siempre es necesario que esto sea así, pero permite generar narrativas más complejas apoyándose en estos puzles.

Es muy importante crearlos con cuidado, evitando que se requiera un evento para cierta acción que va ser la propia que acabe por generar ese evento.

Ejemplo: Untamos un poco de miel en un árbol cercano. Al oso, por muy antropófago que sea, no le amarga un dulce. Al oler la miel nos deja vía libre para continuar. En este caso usamos una combinación objeto-entorno para propiciar el suceso.

• Tiempo

En cierto modo son una evolución del anterior tipo de puzle añadiéndole la dimensión del tiempo. Se generan haciendo que cierta acción del jugador dependa de eventos que ocurren en la aventura, que deben de estar controlados por el tiempo.

La clave de estos puzles es ajustar el tiempo, no se puede exigir al jugador que debe esperar por un evento demasiado tiempo para resolver un puzle que sin ese evento seria imposible de completar. Hay que darle un sentido lógico dentro del mundo creado y razonable para los jugadores y jugadoras.

Ejemplo: Volvamos al anterior. Si bien habíamos conseguido librar el camino, el apetito del oso es muy voraz, lamerá rápidamente la miel. Si no nos damos prisa en pasar vendrá a comernos también a nosotros.

- Repetición de acciones

En ocasiones se requiere la repetición de ciertas acciones para poder conseguir un objetivo. Muchas veces estos retos van asociados a personajes. Este tipo de puzle puede resultar interesante porque obliga al jugador a memorizar ciertos patrones o formas de actuar para saber cuando es momento de actuar.

Ejemplo: Intentamos bajar la colmena que cuelga de la rama sacudiendo el tronco. Tenemos que repetir varias veces la operación para que caiga. A través de una creciente caída de hojas a cada intento daríamos a entender a los jugadores que algo está pasando.

• Aprendizaje y habilidades

Se pueden utilizar para añadir cierto progreso del personaje a lo largo de la historia, ya sea dándole una razón narrativa o jugable.

- Aprendizaje/memorización

Se puede aportar cierta información al jugador que deba recordar pasado un tiempo para poder resolver algun reto. Es importante informar al jugador de cual es la información que debe memorizar,

puesto que muchos datos serán irrelevantes y no se le puede obligar a recordar todo por si acaso algo es útil en el futuro.

Al usar este reto suele resultar más interesante hacer énfasis en el que debe memorizar el jugador por encima de obligarle a memorizarlo a lo largo de mucho tiempo. Se pueden usar melodías o patrones que obliguen a generar dinámicas en el jugador para el aprendizaje de esta información.

Ejemplos:

- Aprendizaje: nos las arreglamos para entrar en un circo con la intención de que allí nos enseñen cómo domar bestias, así podremos aplicar esos conocimientos con el oso.

- Memorización: el circo es un mundo cerrado. Sabemos que se imparten clases para futuros domadores pero el director del circo, el que tiene que autorizar cualquier ingreso, no recibe a extraños. Agachados, vemos que para entrar en el bungaló del director hay que llamar a la puerta con el puño reproduciendo cierta secuencia. La aprendemos y, una vez que tenemos vía libre, la repetimos para ya dentro tratar de convencer al director de que nos deje entrar a formar parte del circo.

- Habilidades y sentidos innatos

Puede resultar útil en juegos en los que se de la opción de controlar a varios jugadores. Muchos retos que se proponen requieren de cierta habilidad con la que solo cuenta uno de los personajes jugables, por lo cual el objetivo del jugador debe ser llevar a ese personaje hasta el reto que requiere de sus habilidades. También puede ser que la habilidad sea innata gracias a resolver algún reto o a memorizar alguna información, como lo explicado anteriormente.

Ejemplos:

- Habilidad innata: aunque nuestro protagonista no es suficientemente ágil como para subirse a un árbol e ir pasando de rama en rama para superar al oso, en un momento dado podemos controlar a otro personaje para el que no supone un reto.

- Adquirida: aplicamos lo que se nos enseñó en el circo en el ejemplo del tipo anterior para poder superar al oso sin que nos coma en el intento. Resulta más interesante que, en lugar de que nuestro personaje haga unos pases con la mano para dejar al oso a su merced con solo enfrentarlo después del aprendizaje en el circo, tengamos que aplicar ciertas pautas aprendidas, presentándose incluso alguna situación inesperada que nos obligue a improvisar sobre la marcha.

• Secuenciales y combinatorios

De nuevo este tipo de puzle puede servir como objetivo de un nivel entero. En ocasiones requerirá de varios personajes actuando con habilidades, para que mientras uno genera un evento que forma parte de una secuencia de la aventura el otro mediante combinaciones consigue completar un puzle que de otra forma seria imposible. Son conocidos como de tipo maquinaria, porque requiere de una secuencia a la par que se van combinando ítems y objetos para generar una secuencia más compleja.

Ejemplo: Tenemos el control de dos personajes. Excavamos un agujero grande en el bosque, lo tapamos con una tabla amarrada a una cuerda y cubrimos todo con hojas. Uno de los personajes, pequeño y rápido, hace de reclamo mientras el otro, grande y fuerte, espera escondido sosteniendo la cuerda. El oso viene persiguiendo al primero de ellos, el humano pasa por encima de la tabla y, cuando llega la bestia, el segunda tira de la cuerda para hacer que se caiga en el agujero. Así logramos despejar el camino.

• **Metapuzles**

Esta categoría es especial y única de cada juego, se ha explicado brevemente en los demás tipos de puzles. La clave esta en generar retos que requieran del conocimiento por parte del jugador de la lógica que sigue el mundo de la aventura, ya que en el mundo real pueden carecer de sentido. Pueden resultar muy útiles para conseguir que el jugador se sienta parte de lo que se esta contando ya que le hacen identificarse de una manera directa con el personaje principal. Se van a citar tipos concretos ya que categorizar estos puzles resultaría imposible.

- ¡Serán cabrones!

Se basa en abusar de la posición de poder que tiene el diseñador sobre el jugador situando un objetivo inalcanzable que este separado de la secuencia principal. Según se avance en la aventura el jugadora pensara que hay una tarea pendiente por hacer, hasta que se de cuenta de que no era mas que un engaño y exclamen: “¡Serán cabrones!” debido al enfado por perder tiempo.

Es recomendable no abusar de estos puzles o el jugador o jugadora se terminará por sentir engañado y abandonará la aventura. Lo mejor es reducirlos a un chiste, pero no convertirlos en retos relevantes dentro de la trama de la historia.

Ejemplo: Vamos andando por una cueva, sobresale de una pared una de las aristas de un colosal diamante. Como estamos educados en una sociedad capitalista, los jugadores principiantes intentarán cogerlo. Los aventureros también, pero con la intención de usarlo en un futuro para cortar un cristal o algo similar, como ya han hecho en juegos anteriores (sí, vale, y por la codicia). Pero resulta imposible, no somos capaces de sacar de allí el diamante que no necesitamos. Perderemos un montón de tiempo sin razón: ¡serán cabrones!

• ¡Seré estúpido!

Este puzle es lo contrario que el anterior, la idea es que su solución sea lo mas obvia y sencilla posible, pero mientras el jugador no caiga en la cuenta supondrá un reto incluso mayor que cualquier otro que se plantee. No seria conveniente colocar este reto al inicio sino una vez el jugador acepte las reglas del mundo y lleve un bagaje detrás para que sus pensamientos estén más corrompidos y sea más fácil engañarle.

Una vez más no hay que convertir la broma en algo pesado, no se puede usar este tipo de puzles como pretexto para engañar colocando demasiados objetos alrededor para perder al jugador. En ocasiones estos puzles no saldrán bien y el jugador lo pasara por encima asique tampoco se debe abusar de ellos.

Ejemplo: Al lado de una puerta hay una llave colgada en la pared, inalcanzable sin ayuda. El jugador es muy probable que asocie la llave a la puerta cerrada e intente conseguirla por todos los medios antes de que se le pase por la cabeza que la puerta puede o no tener cerradura, o que ésta pueda estar sin echar.

• ¡Qué mamones!

Este puzle requiere de una justificación narrativa, en esencia es usar algún puzle complejo del que se haya dado ha entender que la solución será una para que acabe por ser otra totalmente diferente pero que tenga mucho más sentido dentro de la historia del juego. Solo es recomendable cuando se tenga una idea que requiera una resolución muy sencilla para evitar que el jugador desista de completar un puzle que considere inútil para su propósito final.

Ejemplo: Un hombre lleva una cartera en el bolsillo de atrás, a punto de caérsele, con un montón de billetes sobresaliendo, invitando a ser robada. Además, se detiene delante de nosotros en un puesto del mercado, se agacha y así la deja todavía más visible y accesible. El jugador, que hasta ese momento no había necesitado dinero para nada, ni tenía motivo para pensar que sí lo precisaría en el futuro, acabará por coger la cartera porque le resulta evidente que lo están invitando a hacerlo.

• Exógenos

Esta categoría aparece en la mayoría de las aventuras gráficas y consiste en introducir elementos ajenos a la aventura. Esto no quiere decir que no tengan ningún sentido lógico o que sean algo extraño para el jugador, pero suelen ser construidos en una escena aparte o sobreponiendo un panel sobre el mundo en el que esta el personaje para proponer un reto aparte de los ya presentados anteriormente. Es una buena practica para hacer descansar al jugador del reto principal y que se transforme a un entorno aparte que le presente un nuevo reto pero que se resuelve de una manera mucho más directa.

No se van a dar ejemplos concretos puesto que es imposible ejemplificar todos, ya que hasta se pueden crear específicamente para un juego en concreto. Pero se hablara de los tipos mas comunes para una mayor comprensión de este tipo de puzles:

• Laberintos

Son un tipo de puzles exógenos difícil de justificar dentro de la narrativa y en ocasiones puede llegar a resultar aburrido por su poca innovación. El objetivo es mover algún tipo de elemento que se mueve por un mapa hasta llegar a la salida, puede resultar hasta excesivamente sencillo de resolver si no se crea correctamente.

• Sliders

Se suelen usar para crear puzles en el sentido literal. La idea es dar un elemento, generalmente una imagen, descolocado y que mediante huecos que queden libres el jugador deba mover esos elementos hasta hacerlos encajar correctamente

• Acción, arcade, plataformas…

Cada vez es común introducir elementos de otros géneros de la aventura dentro de las aventuras gráficas, aunque no suele ser recomendable. Más allá de los problemas de integrar este otro genero dentro del marco de los controles definidos para una aventura grafica suele ser complicado que resulten divertidos para la mayor parte de los jugadores. Esta claro que el publico este tipo de juegos mas narrativos y basados en otro tipo de retos buscan enfrentarse a algo diferente mas unido a una capacidad mental y deducción. La calve para introducir este elemento pasaría por no introducir una mecánica muy compleja, algo que los buenos y experimentados jugadores de este género, que fundieron sus cartuchos de Mario y Sonic, rápidos de reflejos y capaces de calibrar los obstáculos de manera instantánea, los superarían con un bostezo. Pero al no ser este el objetivo de las aventuras graficas es posible que estos retos lleguen a frustrar y fastidiar al público de estas aventuras.

• Juegos de mesa

Es un nuevo tipo, más complejo, de puzles exógenos. El problema de este tipo de puzles recae en que se obliga a los jugadores y jugadoras a aprender nuevas mecánicas y generar dinámicas totalmente diferentes a las usadas en el resto de la aventura. Esto puede generar que una gran obra de aventuras graficas sea abandonada por requerir conocer cierto juego de mesa implementado en algún nivel. Puede resultar interesante introducir estos puzles haciendo uso de las mecánicas ya existentes, aunque esto los vuelva más complejos de primeras es el propio diseñador el que decide y nivela la dificultad del puzle y puede simplificarlo pero permitir controlarlo haciendo uso de unas mecánicas ya presentadas al jugador, a continuación se nombran algunos ejemplos:

• Cuadrados mágicos, criptogramas, sudokus, sopas de letras…

La clave de este tipo de puzles es que no saquen al jugador de la inmersión y de la narrativa creada. Al fin y al cabo las aventuras graficas son una forma de contar historias y ninguno de los puzles y retos que se propongan deben de destruir esta base.

Es muy común usar lenguaje ambiguo como se explico en las recetas, haciendo pasar ciertos objetos por otros o dobles sentidos para dar mayor complejidad a los puzles, pero un grado superior de satisfacción al jugador y jugadora a la hora de resolverlo. También la observación debe de ser una dinámica desarrollada por los jugadores, porque al fin y al cabo se podría hace el símil como si fuesen detectives que necesitan encontrar pistas y usarlas. Por esto muchas veces se pueden camuflar los elementos o esconder dentro de otros para obligar al jugador a ser concienzudo en sus dinámicas.

• Arbitrarios

Se recomienda huir de ellos, pese que se han hablado de los metapuzles y de que en ocasiones se puede imponer la lógica del mundo ficticio creado al real para generar ciertos puzles, no hay que caer en puzles que sean completamente arbitrarios y que se resuelvan a base de que el jugador o jugadora tengan un golpe de suerte. Se recomienda huir de ellos.

Ejemplos:

• Necesitamos comprar esa miel, pero no tenemos dinero ni nada que intercambiar. Recogemos unas plantas en el bosque sin saber para qué nos van a servir, solo porque se pueden coger. Ese hecho, sin ninguna relación, provoca que la tendera abandone el puesto para ir al servicio. Entonces podremos robar un tarro.

• Para ganarse la confianza de un personaje, el jugador tiene que responder adecuadamente a un cuestionario. Sí existe una relación entre las posibles respuestas que se muestran en pantalla y esas preguntas, lo que hace que se pueda resolver, aunque no por ello deje de ser arbitrario. Pongamos que se tiene que elegir el número de respuesta en función de la cantidad de nexos que lleve la pregunta. Esta en su redacción utiliza tres, así que la tercera opción será la correcta. Aunque siga un patrón, que el jugador puede acabar descifrando, es un rompecabezas arbitrario porque no utiliza las leyes o reglas del universo introducido.

Tras la clasificación de los puzles puede ser que los jugadores mas veteranos de este genero de juegos piensen que hay algunos no definidos como los puzles conversacionales. Pero muchos de estos puzles comunes no forman una categoría propia, sino que se enmarcan dentro de una ya explicada o son una combinación de varias. Una ves se han propuesto todos los puzles que pueden constar en una aventura grafica se debe tener en cuenta que deben seguir un funcionamiento general que se explica a continuación. Mas adelante se explica la propuesta hecha para el primer nivel del juego que se va a implementar el cual esta constituido por muchos de los puzles y retos ya descritos.

Una vez hecha esta clasificación según los tipos de objetos que intervienen y el funcionamiento se hace una división de los tipos de retos según su estructura. Los puzles de una aventura no funcionan de forma independiente o aislada, sino que se estructuran en conjuntos de puzles que siguen una jerarquía y unas relaciones. Hay tres tipos generales de estructura:

• Lineales

El jugador tiene solo un puzle, subpuzle o secuencia de puzle a resolver en cada momento. Un puzle habilita otro puzle y así sucesivamente.

• En paralelo.

Al jugador se le presentan varias secuencias de puzles a resolver en un mismo tramo de partida.

• Bifurcativas

La solución de un puzle habilita nuevas acciones, pero DESCARTA otras. También podemos incluir aquí los puzles optativos.

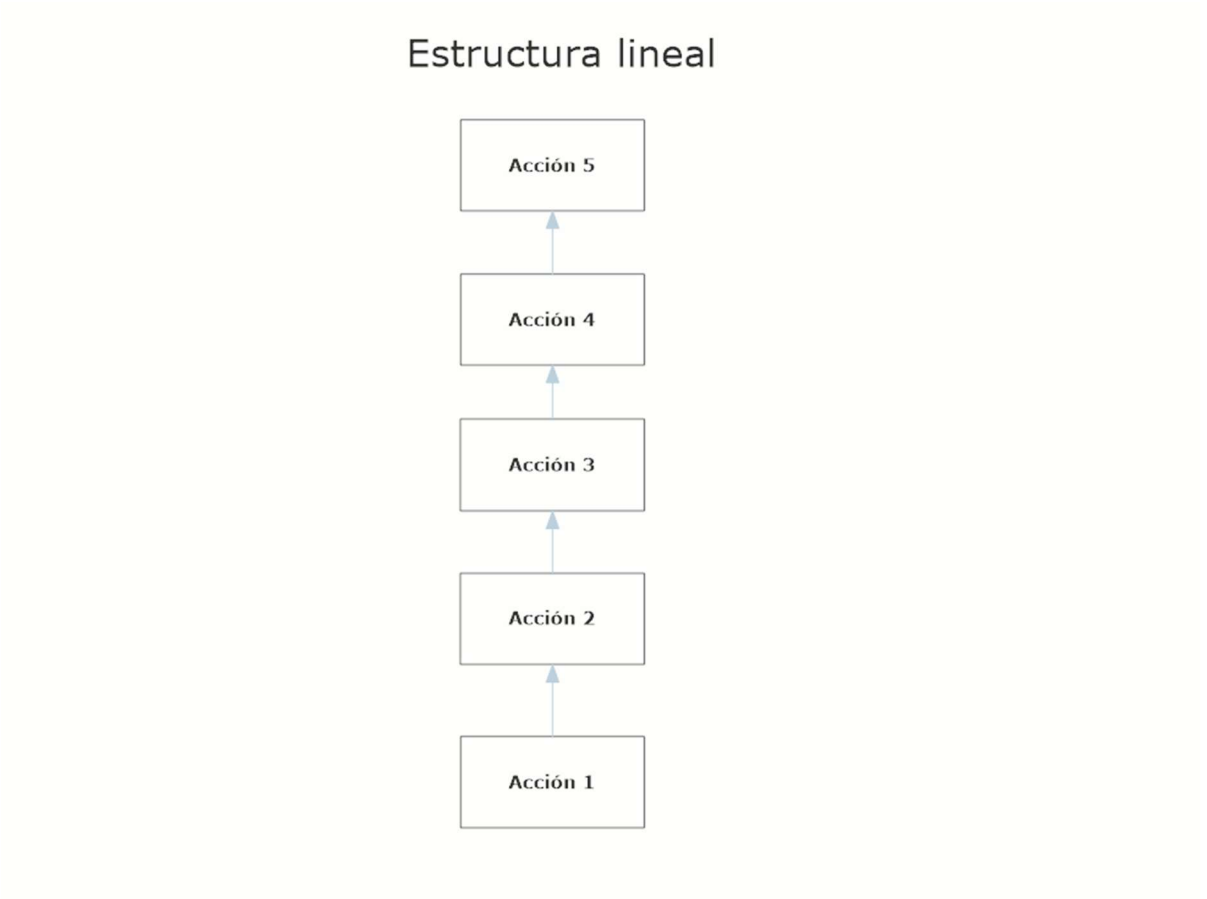

Figura 5: Clasificación de los retos según su estructura

Es importante tener en cuenta que no solo hay que definir los tipos de puzle sino las características propias de ellos que estarán definidas en gran parte por el público objetivo. No se puede ser excesivamente técnico en este aspecto, pero si se continúa siguiendo las indicaciones de R. Fervenza los principales atributos a valorar en un puzle son:

• Dificultad.

Es una tarea compleja medirla debido a que va unida a la propia naturaleza del puzle. Influyen muchos valores en ella como el momento de la partida, la información con la que cuenta el jugador o a quien va dirigido, lo que es difícil para unos puede ser fácil para otros. Pero se puede caracterizar por:

• Profundidad.

La profundidad depende del numero de pasos que requiere el jugador para resolver cierto puzle. Este parámetro por si solo no significa nada puesto que los pasos pueden ser muy sencillos, pero es necesario encontrar un balanceo con el resto de características de la dificultad.

• Exhaustividad.

Define el grado de atención que se debe prestar para resolver un puzle. Si requiere que el jugador memorice algo o aplique una lógica demasiado compleja es recomendable no extender el puzle excesivamente en el tiempo puesto que puede acabar por agobiar al jugador o jugadora.

• Sutilidad.

Puesto que muchos puzles se basan en la combinación de otros elementos, muchas veces estas relaciones conceptuales pueden ser mas o menos sutiles o evidentes. Esta característica es muy complicada de medir debido a que depende demasiado de la experiencia del diseñador de los retos y del jugador, además de otros factores culturales. El problema es que en ocasiones los diseñadores es normal que cuenten con mucho bagaje dentro de este tipo de juegos e implemente puzles muy poco evidentes. La calve para esta característica recae en las pruebas con usuarios que no formen parte del desarrollo y ver como se desenvuelven para con la información obtenida se pueda balancear de la manera correcta la dificultad.

• Curva de dificultad.

Este factor viene determinado por el avence del jugador o jugadora por la aventura, no serán iguales los retos propuestos al inicio que en el ultimo nivel, la clave recae en que la dificultad aumente lo suficiente para no aburrir al jugador, pero no tanto como para crear un reto

imposible. El objetivo es conseguir mantener la dificultad dentro de esa zona ideal llamada *Flow Zone*.

• Variedad.

Para los jugadores la variedad es un factor a tener en cuenta, ya que esto evita en muchas ocasiones que los juegos acaben llevando al aburrimiento. El problema es que para los desarrolladores la variedad supone crear más y más elementos que pueden no llegar a tener un limite. Por ello hay que llegar a un punto medio en el que el jugador y jugadora estén contentos a la vez que los desarrolladores no se han vuelto locos por el camino.

• Integración.

Es un parámetro muy importante partiendo de la base ya explicada de que la narrativa es un aspecto fundamental de estos juegos. Es obligatorio encontrar la manera de integrar todos los restos y puzles que se propongan dentro del mundo que se ha creado para que el jugador no lo sienta como algo impostado y se pierda la magia de la inmersión que existe en las obras ficticias.

3.2. Modelo de negocio

El modelo de negocio que adopta el videojuego, o el nivel creado, es plenamente gratuito. Al tratarse más de una demo que muestre todas las mecánicas del juego y presente elementos importantes no es posible tratar de rentabilizarlo, sino utilizarlo de una forma publicitaria o promocional para dar a conocer el trabajo hecho durante el desarrollo. Aun con todo esto una vez finalizado el proyecto al completo y no solo como una demo tampoco se considera un producto monetizable.

Su modelo de negocio finalmente se orienta al *Free-to-Play*, usando este modelo las empresas obtienen beneficios a partir de la publicidad o venta de algún producto virtual dentro del mundo del videojuego. En este caso, al tratarse de un proyecto con un origen dentro del ámbito académico, el objetivo no será ganar dinero a partir de la publicidad o un mercado virtual sino simplemente aprender a utilizar las herramientas necesarias para crear un proyecto completo desde cero a la vez de mostrar todas las capacidades de desarrollo e implementación.

4. Diseño

En este capítulo se comienza a explicar el diseño del juego, hablando de todos los *Assets* utilizados como base y de que forma se ha desarrollado la funcionalidad que vaya asociada a estos recursos. En este punto es importante recordar la forma de implementación seguida mediante la metodología de trabajo SCRUM, pero para una mayor claridad a la hora de exponerla se explica la versión final de las implementaciones de todas las funcionalidades, a las cuales se ha llegado siempre implementando una base y mediante iteraciones y pruebas con posibles usuarios finales mejorándola hasta el resultado final.

4.1. Arquitectura general de la aplicación/sistema/servicio

La estructura seguida en las funcionalidades implementadas está muy basada en el tipo de juego que se busca crear. La gran potencia de Unity a la hora de diseñar e implementar las diferentes funcionalidades se basa en el concepto de los *Componentes* los cuales pueden ser diferentes funcionalidades, ya creadas por Unity o creadas por el usuario, que se añaden a los *GameObject* de la escena. El principal de estos *Componentes* es el *transform* que define que lugar ocupa un objeto en una escena, su escala y algunas propiedades que tienen la mayoría de los *GameObject*, aunque también existen otros como añadirle una malla 3D al *GameObject*, una fuente de audio o un *MonoBehaviour* que esto es un script en C# totalmente programable por el desarrollador haciendo uso de la API de Unity para implementar todas las funcionalidades únicas que desee en sus juegos. De aquí en adelante para hablar de las funcionalidades se tiene en cuenta que más tarde, en la implementación, se traducen a *MonoBehaviour* que contienen el código en C# necesario para implementar las funcionalidades descritas.

Lo primero y más importante es generar un personaje que se mueva en el mundo que más adelante se diseñara y creara. Para esto se deben capturar las entradas del usuario, teclado/ratón o mandos, y a partir de ellas generar movimiento en el *GameObject*, a esto también se le añade animaciones haciendo uso del motor de animaciones de Unity. Una vez hecho esto se implementan las mecánicas principales basadas en los objetos con los que el jugador o jugador pueden interactuar en la escena. Estos objetos tendrán diferentes funcionalidades asociadas como lo puede ser usarlos, cogerlos, hablar con ellos o mirarlos. Con lo cual lo principal es crear la funcionalidad básica de objeto en la escena con el que pueda interactuar el jugador y a partir de ahí asociarle la funcionalidad que realiza cuando se interactúa con él.

Otro concepto importante es el inventario, los objetos interactivos que tengan coger como funcionalidad asociada a interactuar con ellos, deben añadir un objeto al inventario cuando el jugador se acerque a ellos e interactúe. Para esto se implementa la funcionalidad de coger añadiéndole a esta

funcionalidad un ítem el cual contendrá la información que tendrá ese objeto al añadirlo al inventario. De esta forma cuando se colocan diferentes objetos interactuables con la funcionalidad de coger se debe añadir también toda la información que va a tener el ítem cuando el jugador lo recoja y lleve a su inventario. La información que contendrá el ítem será propiedades como la imagen que muestra una vez está en el inventario, un nombre del objeto y los usos que podrá hacer el jugador con el ítem.

También es necesario implementar la funcionalidad de uso para elementos con los que el jugador pueda interactuar, como una puerta que se puede abrir o un interruptor que puede pulsar. Para esta funcionalidad se crea una interfaz base para después poder crear cada funcionalidad apoyada en la base, y así no sea necesario crear de cero cada funcionalidad.

Para la funcionalidad de hablar es necesario crear una estructura de datos con forma arbórea que contenga el texto asociado a una opción de diálogo y una lista de hijos que contengan un subárbol de diálogos. De esta forma hasta que el jugador no le pregunte a que se dedica al *NPC* (Personaje No Jugable) no se abrirán las opciones de diálogo para preguntarle más en profundidad sobre el trabajo, por ejemplo. Esta funcionalidad, aparte de tener todos los datos asociados al árbol de diálogo, debe abrir paneles de la interfaz gráfica para que el usuario pueda elegir entre todas las opciones y habilitar que la última opción salga de la conversación o vuelva al anterior nivel del árbol de conversaciones, en caso de que sea el nodo padre del árbol o sean nodos hijos respectivamente.

En definitiva, para implementar los objetos interactivos y sus funcionalidades asociadas se crea una base que sea totalmente editable desde el motor de Unity. Esto se traduce en una carga mayor de trabajo a la hora de implementar al inicio del proyecto, pero una vez implementada toda la base tan solo es necesario diseñar los niveles y retos que se vayan a proponer al usuario, y desde Unity rellenar todo lo necesario reduciendo la implementación a funcionalidades concretas o puzles que se desee implementar.

4.2. Arquitectura de la información y diagramas de navegación

A continuación, se muestran figuras que contienen las arquitecturas de las funcionalidades básicas explicadas como lo son:

- *InteractiveObjects*, para los objetos con los que el jugador o jugadora puede interactuar.

- *Interaction*, función base de la que heredan todas las posibles funciones de los objetos que se explican a continuación.

- *CatchInteraction*, hereda de *Interaction* e implementa la funcionalidad de coger un objeto y añadir el Item asociado al inventario.

- *UseInteraction*, hereda de *Interaction* e implementa funcionalidades personalizadas de cada objeto, como por ejemplo interruptores, puertas entre otros.

- *TalkInteraction*, hereda de *Interaction* e implementa la funcionalidad de hablar y abrir el menú de conversación, además de contener una estructura de datos arbórea que contiene la información del árbol de diálogos.

- *LookInteraction*, hereda de *Interaction* e implementa la funcionalidad más básica que es mirar un objeto para que el personaje diga algún diálogo que pueda dar pistas al jugador o jugadora.

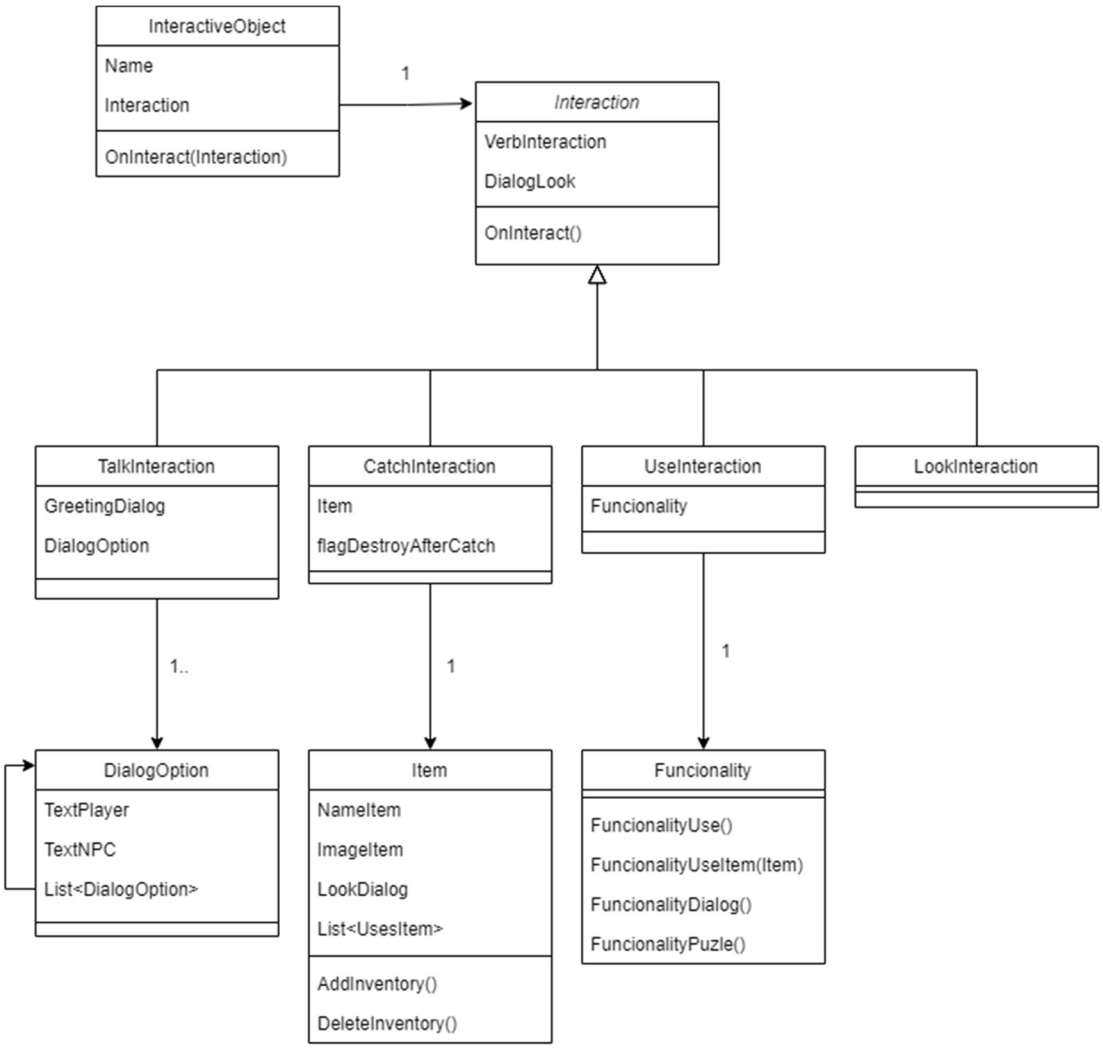

Figura 6: Diagrama de las clases de funcionalidad de los objetos interactivos y sus interacciones

Como se ve cada clase contiene la información necesaria para mostrar al jugador como el nombre de los objetos o ítems y sus funciones asociadas. Los ítems tienen una lista de usos, estos usos pueden ser o con otros ítems, provocando así la mezcla de objetos dentro del inventario para obtener otros que nos permita resolver otros puzles; o se pueden usar con objetos interactivos de la escena, por ejemplo, darle algo algún personaje o usar un destornillador para abrir algo con tornillos.

También se ve que la clase *Funcionality* no es más que la base de posibles funcionalidades. Una de ellas es cuando el jugador interactúa con el objeto y activa la funcionalidad, *FuncionalityUse*, pero también existen otras:

- *FuncionalityUseItem*, esta funcionalidad es llamada cuando se combina un item desde el inventario con un objeto que tiene esta funcionalidad asociada, y de esta forma se comprobaría si el Ítem es el correcto y se ejecutaría la acción. Por ejemplo, hasta que no se use una llave con una puerta cerrada esta no se abrirá.

- *FuncionalityDialog*, esta funcionalidad se usa cuando el usuario clica en un diálogo con uno de los personajes de la escena, de esta forma se permite bloquear ciertas funcionalidades hasta que el jugador hable de algo con algún personaje. Por ejemplo, si hay un personaje que controla la luz de una sala, hasta que el jugador no le pida al personaje que encienda la luz, clicando en la opción de diálogo que lo diga dentro de la conversación, no se ejecutará la función de encender la luz.

- *FuncionalityPuzle*, esta acción se ejecuta cuando el jugador finaliza un puzle correctamente. Por ejemplo, si hay una pared secreta oculta y no se quiere que se abra hasta que el jugador complete un puzle propuesto.

4.3. Flujo lógico del nivel implementado

El proyecto se basa en implementar toda la lógica y funcionalidad sobre un nivel que debe contener retos y puzles ya explicados. Para ello se ha diseñado todos los retos que deberá resolver el jugador para conseguir su objetivo final del nivel, este diseño tiene una estructura arbórea, puesto que para crear estos retos y su flujo de ejecución lo mejor es partir del objetivo final que se le propone al jugador y de ahí crear sub-retos y como se resuelven estos de la misma forma que el principal. Así hasta llegar a los objetivos reales inicialmente del jugador que son acciones básicas como coger cierto objeto, hablar con cierta persona o usar algún objeto de la escena o el inventario.

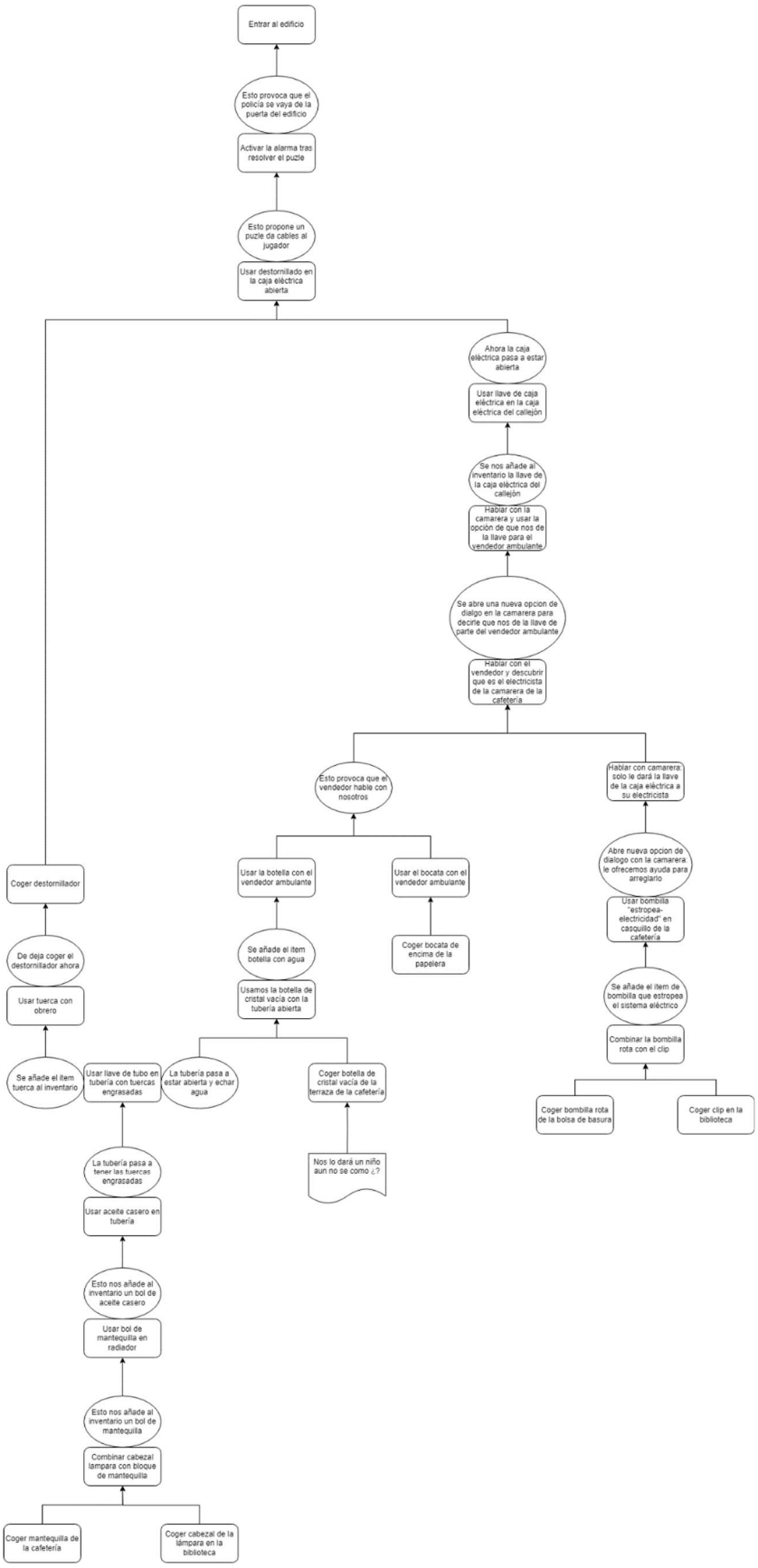

Figura 7: Flujo de retos propuestos al jugador en el nivel 1

4.4. Diseño gráfico e interfaces

A la hora de comenzar la implementación de los niveles es necesario definir ciertos parámetros artísticos y gráficos para saber que trabajo deben hacer los artistas y modeladores, o en su defecto como en este caso que recursos se deben buscar siguiendo dichos parámetros. Además, es importante nombrar ciertas fuentes de inspiración porque crear algo nuevo requiere de un equipo en el ámbito gráfico del que este proyecto no se dispone. En cuanto a colores e iluminación una posible referencia es “Indiana Jones”, con los colores cálidos que la identifican con tonos ocres y amarillentos. Luego en cuanto al estilo artístico y de tono otro referente, ya dentro del mundo de los videojuegos, es “Broken Sword”, pese a que dentro de la saga tiene títulos en 2D y 3D el objetivo es recrear, mediante un estilo diferente y propio, esa atmosfera de conspiración de la antigüedad que genera con su estilo artístico o elementos de las interfaces.

4.4.1. Estilos

El estilo artístico de un videojuego es un elemento muy importante a la hora de dar el producto a conocer y de que sea más fácil que a los posibles usuarios les entre por los ojos y eso ayude a que quieran probarlo. Esto no significa que todos los videojuegos deban tener unos estándares de los videojuegos triple A, con modelos 3D, texturas e iluminación hiperrealista. Un estilo artístico innovador y rompedor puede ser igual de válido o incluso superior a juegos que busquen el hiperrealismo.
Existen casos como por ejemplo “Fornite”, que estando dentro de un género donde otros juegos apostaban por estilos mucho más realistas, conquistó al público gracias a sus gráficos *cartoon* y más distendidos. En el ámbito de los juegos indie también se pueden nombrar juegos como “Journey” en el cual todo su potencial recae en su exquisito estilo gráfico.

Dentro del género de las aventuras gráficas, que son el principal referente de este proyecto, se ha visto que los usuarios no son muy favorables a gráficos muy realistas, sino más bien lo contrario como se ven en títulos como “Broken Sword” o “Monkey Island”, que al tratarse de juegos en 2D prevalece estilos artísticos más enfocados en ser visualmente atractivos. En la siguiente figura podemos ver el estilo artístico de una de las últimas aventuras gráficas clásicas que ha sido lanzada al mercado, “Return To Monkey Island”.

Figura 8: Estilo artístico de “Return to Monkey Island”

Vemos que este estilo se aleja mucho de cualquier otra obra, y pese a que hay una parte de los fans de la saga a los que no les ha convencido, hay otra gran parte a la que sí les ha gustado este lavado de cara, y además el estilo artístico consigue su objetivo que es ser identificable y que la gente al pensar en él piense automáticamente en este videojuego.

En el caso de este proyecto no se dispone de plena libertad para crear un estilo nuevo ya que en el apartado de los *assets* visuales se van a coger modelos ya creados, pero pese a ello no se busca un estilo realista. El objetivo es aprovechar el 3D en lo que a mecánicas de exploración e interacción se refiere, pero tratando de que su estilo visual sea más identificable con juegos más *cartoon*, que no por ello debe ser una obra humorística. Se va a buscar un balance entre los colores, iluminación y el estilo artístico para llegar a un resultado que no busque ser realista pero que si dote al juego de una seriedad suficiente para que el jugador o jugadora no se lo tome como una obra de humor, pero sin perder ese ambiente que se crean en muchas aventuras como “Broken Sword” o “” entre otras. Un estilo demasiado serio u oscuro en ocasiones puede terminar por agobiar al usuario y requiere una narrativa que esté en total sintonía con el tono, y no se considera que en este caso sea así.

4.5. Lenguajes de programación y APIs utilizados

Para la implementación se va a utilizar Unity como motor principal, en la cual se utilizarán todas las herramientas de las que dispone como lo son: su motor gráfico, el motor de animaciones, su motor de sonido y los componentes que permiten añadir funcionalidades a los objetos de Unity. Este componente de scripts pone a disposición de los desarrolladores una API para facilitar la programación e implementación de funcionalidades extras de las que no dispone Unity. Todo el proyecto se desarrolla íntegramente en la versión de Unity 2022.1.23f1, que es la versión más reciente LTS para

evitar fallos y bugs durante el proyecto. Además, Unity se ejecuta en un ordenador con Windows 11 como sistema operativo, aunque gracias a la separación de Unity con el sistema operativo se podrá exportar el proyecto a otros sistemas operativos como lo pueden ser Linux o Android entre otros.

Esta elección es debida a que los conocimientos a la hora de desarrollar en Unity son más amplios que en otros motores como lo pueden ser Unreal Engine o Godot, además se ha seleccionado la versión de Unity más moderna que era LTS (Long-term support) para evitar posibles fallos de versiones posteriores pero que no están totalmente testeadas y probadas. Y por último la elección de Windows 11 es porque el ordenador del que se dispone para realizar el proyecto es Windows 11 y además es el sistema operativo que es más compatible con herramientas necesarias para el desarrollo de un proyecto de un videojuego.

Gracias a usar Unity hay disponible una enorme cantidad de recursos tanto teóricos a la hora de implementar ciertas funcionalidades, como recursos para usar en el videojuego como lo son funcionalidades ya implementadas y modelos 3D que sirven como base para crear escenarios y personajes. Para este segundo propósito el sitio web más conocido y útil es la Unity Asset Store la cual es una tienda online con todo tipo de recursos para crear proyectos por gente que no disponga de habilidades de modelaje o artísticas, como es el caso del proyecto actual, que como ya se ha explicado por falta de conocimientos y tiempos se recurre en varias ocasiones por utilizar recursos ya existentes y limitar la tarea del diseño del escenario y sus elementos artísticos a colocar *assets* ya creados por otros desarrolladores.

5. Implementación

Se comienza por crear una escena vacía en la que poder implementar todas las funcionalidades básicas para luego llevarlas a los escenarios acabados del juego. Esta versión inicial sin ningún componente artístico del juego está construida a partir de las primitivas 3D que da Unity, se llama prototipo y sirve para probar toda la lógica del juego sin tener que preocuparse de gastar recursos en crear escenarios, que luego debido a fallos en la base de la idea no se llegarían crear, ya que en ocasiones al probar las mecánicas en el prototipo surgen nuevas ideas y se descartan otras.

Para crear todas las funcionalidades se sigue un proceso común. Se crea un *GameObject* vacío, el cual contiene un *transform* que es importante si queremos que esa funcionalidad tenga sentido dentro del mundo virtual que se crea, por ejemplo el movimiento del personaje si es importante y su colocación dentro de la escena, mientras que la funcionalidad de gestionar el inventario no importa el lugar del mundo virtual que ocupe mientras que sirva para interactuar con objetos y el inventario. Una vez está creado el *GameObject* se le comienza a añadir los componentes que son necesarios para que tengan sentido dentro del juego, el personaje además de la funcionalidad de movimiento debe tener un controlador para las animaciones del avatar, la propia malla 3D que contenga el avatar, una fuente de sonido para reproducir los diálogos y demás sonidos que emita el protagonista. Y por último a los *GameObject* se le añade un script C# en el que se implemente todas las funcionalidades necesarias, ya explicadas en el apartado de diseño y desarrolladas más adelante. Tras tener ya el *GameObject* con toda la lógica implementada se ha creado un *Prefab*, para poderlo guardar y replicar siempre que se quiera simplemente arrastrando, gracias al *Prefab* los cambios que hagamos en una instancia del mismo afectarán a todas las demás, y así no se tendrá que cambiar a cada uno los parámetros.

5.1. Movimiento Personaje y cámara

Para el movimiento se hace uso de un componente ya existente de Unity llamado *CharacterController*. Este componente aporta llamadas desde el script, que se ha creado para el movimiento del personaje, que permiten moverlo por la escena de una manera más realista y sencilla que si se implementara el movimiento desde cero. Para el movimiento se ha hecho uso del nuevo módulo de entradas de Unity, llamado *Input System*, el cual se puede configurar para invocar eventos de código cuando se pulse un botón. De esta forma siempre que se desee configurar los controles se hará de esta forma. A la vez el movimiento del personaje va asociado a un modelo 3D de un personaje *riggeado* al que se pueda añadir animaciones, para esta funcionalidad hay que usar el *AnimatorController* que permite diseñar una máquina de estados con los diferentes clips de animación, por los que se irá moviendo en función de unos parámetros que se *setean* desde código. Por ejemplo, para la animación de andar hay un numero de 0 a 1 que lo controla, cuando vía código se detecte que la entrada del control de avanzar es

mayor que cero se comenzará a incrementar la variable hasta que llegue a la velocidad máxima de movimiento.

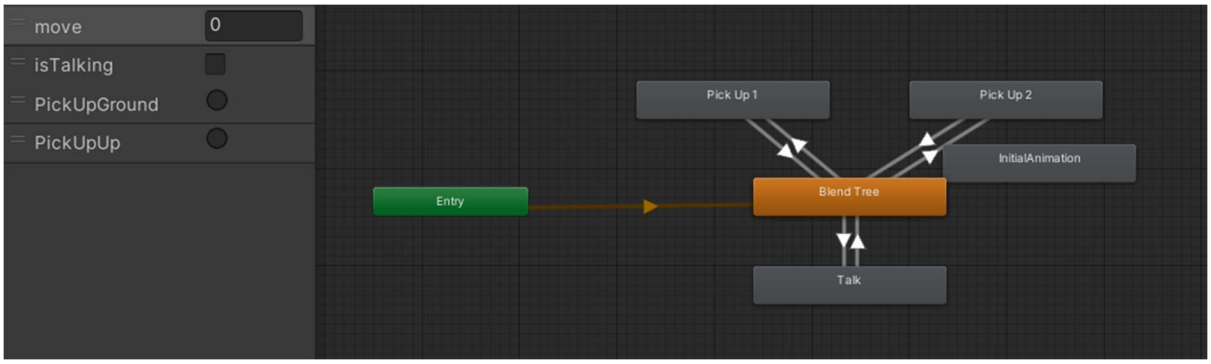

Figura 9: Máquina de estados del avatar de la protagonista

Como se ve en la figura para el movimiento se ha creado un *Blend Tree*, el cual permite en un slider colocar diferentes animaciones, y según aumente el valor de ese slider se irán mezclando las animaciones dando un efecto más realista.

En cuanto a la cámara se ha usado *Cinemachine* que es una herramienta muy útil en Unity para la gestión de cámaras, tanto en las animaciones como para las cámaras de juego. Esta herramienta aporta un cerebro para la cámara que está preconfigurado para juegos en tercera persona con vista libre. Esta cámara habrá que configurarla a la distancia que queremos del avatar, bloquear hasta donde se puede mover, y decir que *GameObject* de la escena es el que debe seguir.

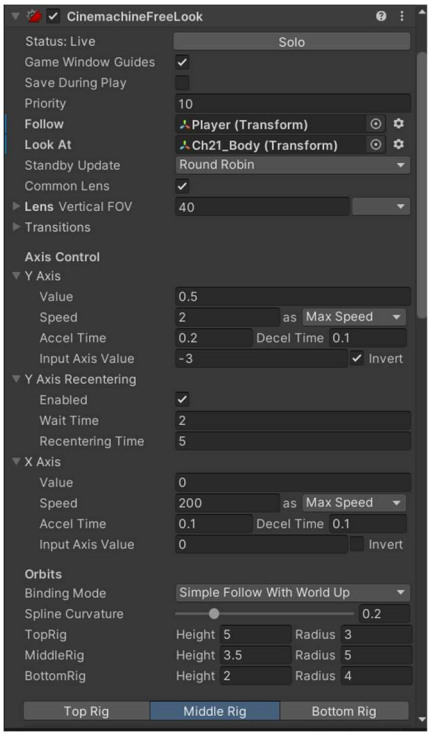

Figura 10: Configuración cámara FreeLook de Cinemachine

5.2. Objetos Interactivos

Esta funcionalidad es la más importante dentro del tipo de juego que se está creando, la lógica a implementar está explicada en el apartado de diseño. La idea es crear una clase que contenga todos los objetos de la escena con los que el jugador o jugadora puede interactuar. Esta clase gestionará que cuando el jugador se acerque al objeto se marcará como objetivo, y se escribe la interfaz gráfica, el verbo de interacción y el nombre del objeto. A la vez esta clase contiene una interacción, que puede ser de los cuatro tipos ya explicados, y cuando el jugador pulse el control de acción se invoca el evento de acción principal, que en función del tipo de interacción asociado al objeto interactivo hará una función u otra.

Para escribir en la interfaz gráfica, Unity requiere de un *Canvas* el cual es un *GameObject* que está fuera del mundo virtual y tiene las proporciones de una pantalla de un ordenador. Los GameObject que sean hijos del *Canvas* en lugar de tener el componente *Transeform* dispondrán del *RectTransform*, que en esencia es lo mismo pero aplicado a interfaces de usuario. En el *Canvas* se crea un panel en la parte superior que contiene texto y será rellenado con los objetos interactivos, o ítems como se verá más adelante, a los que esté prestando atención la protagonista. De esta forma el usuario sabrá en cada momento sobre que objeto dentro del mundo del juego está realizando las acciones.

Cada tipo de interacción implementa sus propias funciones, se destacan algunas importantes:

- *Catch Interaction*: esta funcionalidad contiene una clase ítem con toda la información del objeto que va a añadir al inventario, se explica más adelante.

- *Use Interaction*: contiene una interfaz base de funcionalidad, la cual declara algunos métodos cuando se usa un ítem del inventario o cuando el jugador o jugadora usan un objeto. Para cada funcionalidad se implementa la interfaz base y se escriben los métodos en función de lo que se busca que haga el objeto.

- *Talk Interaction*: para implementar esta funcionalidad se crea una clase que contiene toda la información del árbol de diálogos. Esta clase tiene una estructura arbórea y contiene la cadena de texto que hace referencia al diálogo que diría el protagonista, que sirve para escribirse en los botones y que los jugadores sepan que opciones tienen para decir. Asociado a este texto de la opción hay una respuesta del NPC con el que se está hablando. Y por último también hay asociada una lista de posibles subopciones dentro del árbol de diálogo. De esta forma cuando se pulse una opción que si tenga la lista asociada cuando se escriban las opciones de nuevo en pantalla el jugador o jugadora habrá accedido a un subárbol de la conversación, y para volver al anterior nivel deberá siempre pulsar la última opción.

5.3. Objetos con funcionalidades asociadas

Se ha creado un comportamiento de Unity base para representar las funcionalidades de un objeto para que sean llamadas por el resto de los componentes creados y tener esa interfaz que puedan invocar, pero que cada funcionalidad la implemente de una forma. Por ejemplo, la llamada al método de activar la función está declarada, pero en el caso de la funcionalidad de una puerta será abrirse, mientras que la funcionalidad de un botón es pulsarse.

Visto esto hay que entender que es necesario crear una base para todas las posibles llamadas de funcionalidades desde el resto de las componentes y objetos, para ello se irán describiendo todas estas llamadas que se han declarado para que luego cada *FuncionalityObject* que se cree las implemente en función de sus necesidades:

- *Funcionalidad:* este método se invoca cuando el jugador interacciona con un *InteractiveObject* que tiene como interacción asociada la *UseInteraction*. De esta forma se pueden crear objetos que con la interacción del usuario ejecuten una función personalizada si se implementa este método en el script que se crea para ese objeto. Por ejemplo, una puerta que sea abre y se cierra cuando el jugador interacciona con ella.

- Funcionalidad Item*:* este método se invoca cuando usamos un item sobre otro *InteractiveObject*, el cual este entre los usos de dicho item. De esta forma usar ítems tendrá un sentido dentro del mundo de la aventura que ejecutaran acciones. Por ejemplo, si usamos una llave sobre una puerta cerrada se debe ejecutar la funcionalidad de abrir la puerta.

- Funcionalidad Dialogo*:* este método es invocado, si se asigna, cuando el jugador o jugador usa una opción de dialogo dentro de una conversación. No es obligatorio que todos los diálogos tengan asociada un comportamiento de funcionamiento, pero en ocasiones es interesante en este tipo de juegos realizar una acción concreta en el momento en el que se usa un dialogo con un personaje, pero no hasta antes. Por ejemplo, hasta que no hablemos con un niño que esta buscando su pelota no preguntaremos al resto de personajes por la pelota del niño, de esta forma en el dialogo que le preguntemos al niño que busca ejecutara la funcionalidad de añadir la opción de dialogo de preguntar por la pelota al resto de personajes.

- Funcionalidad Puzzle*:* este método es llamado desde un puzle cuando es completado con éxito, el objetivo es que se realice la acción asociada a ese puzle. Por ejemplo, si se completa un puzle que era necesario para abrir una puerta, este método se implementara con la funcionalidad asociada que abra la puerta.

- *Funcionalidad Tras Coger:* este método se ejecuta después de que la protagonista recoja un *IteractiveObejct* que tenga asociada una *CatchInteraction*. De esta forma se pueden

implementar acciones personalizadas al recoger un item si es que así se indica mediante una bandera. Esto puede ser útil en situaciones que se desee bloquear ciertas acciones al jugador o jugadora hasta que se coja un objeto en concreto, por ejemplo hasta que el jugador no coja una carta del amante de un personaje no se desbloquea la opción de dialogo de hablar al personaje sobre su romance oculto.

Se han tratado de crear una llamada para todos los posibles casos en los que el diseñador del juego busque que se ejecute una funcionalidad especifica, de tal forma que solo sea necesario crear el comportamiento *Funcionality* e implemente el método concreto. Esta base creada resulta muy útil para agilizar la creación de funcionalidades y dar siempre la opción de total libertad para crear nuevos retos a lo largo de la aventura no habiendo limite, puesto que si se desea crear un nuevo de tipo de interacción tan solo habría que implementarlo en el método de funcionalidad al usar para que cuando el usuario interactúe con el se ejecute.

5.4. Managers

Cada funcionalidad existente durante el *gameplay* tiene un comportamiento que controla las variables y funcionalidades necesarias para el correcto funcionamiento. Este procedimiento se ha seguido para no aunar todo en un único script donde se entremezclen funcionalidades, sino tener cada funcionamiento encapsulado en una misma clase de comportamiento, y así un cambio en una de ellas no afecte a todas las demás. Para entender el funcionamiento lo mejor es explicar cual es el propósito de cada una de ellas.

5.4.1. GameManager

Encargada de controlar el *gameplay* durante el juego. Es desde donde se bloquea el movimiento de la cámara y del personaje para entrar en el inventario y menús. También controla el inicio de la escena, si se está cargando una partida o se comienza desde el principio para mostrar la animación inicial del nivel, si es que existe. Otras funciones que aporta es mostrar o ocultar el inventario o paneles de conversaciones, estas funciones se delegan en sus respectivos managers, pero desde aquí se controla si se muestra o no el cursor, a la vez de si se bloquea o desbloquea el movimiento de cámara y personaje.

5.4.2. InventoryManager

Contiene la funcionalidad necesaria para gestionar correctamente el inventario del jugador o jugadora. La función principal de este script es añadir ítems al inventario, esta función es llamada desde la interacción *CatchInteraction*, que contiene un objeto tipo *Ítem* como ya se ha explicado. De esta forma lo que hace la función es crear en el inventario un nuevo panel que represente el ítem. Las

funcionalidades de este nuevo *GameObject* se explican más adelante. La otra función que se implementa es el evento de uso de ítem, básicamente este evento coge el ítem del inventario que este marcado en ese momento y lo intenta combinar, también se explica más delante de manera más detallada.

5.4.3. UIManager

Este script gestiona la interfaz de usuario que se muestra durante el juego. Básicamente mientras el jugador o jugadora se mueven por la escena, cuando se acerquen a un *InteractiveObjects* lo suficiente en la interfaz se escribirá en un panel de texto su nombre y se pondrá un icono de objetivo sobre el objeto, para que el usuario sepa donde está exactamente. Estas dos variables, texto e icono, son gestionadas por el *UIManager*, por ello las funciones que se implementan en el script sirven para limpiar la interfaz o escribirla al acercarse a un objeto. A su vez cuando el jugador esté en el inventario escribe el nombre de los ítems según va pasando el cursor sobre ellos y además al arrastrar uno de ellos, como se explica más adelante, la interfaz se actualizará indicando por pantalla que lo que hace a continuación es usar el ítem en cuestión con otro, siendo este otro sobre donde se pase el cursor.

Figura 11: Interfaz de usuario en juego

Figura 12: Interfaz de usuario en inventario

5.4.4. DialogsManager

Esta funcionalidad se encarga de gestionar una cola de diálogos de la protagonista o del resto de personajes. Desde este *manager* cada vez que reciba llamadas de solicitar reproducir un nuevo diálogo los añadirá en una cola y según vaya terminando cada diálogo activo se reproduce el siguiente de la cola, así hasta finalizar la cola entera. Además, cuenta con un método de reproducción implementado mediante una corrutina de Unity, que permite ejecutar esta funcionalidad en un hilo paralelo sin bloquear el resto del juego.

La corrutina en cuestión se ejecuta hasta que la cola de diálogo este vacía, y en esta corrutina se reproduce el audio asociado al diálogo y se escribe en pantalla el subtítulo, cuando haya pasado el tiempo que dura el audio se borra el subtítulo y se reproduce el siguiente, así hasta que la cola de diálogos este vacía. Para la reproducción del audio del diálogo se explica a continuación el manager específico para la tarea.

- *AudioManager:* funcionalidad asociada a la reproducción de cualquier sonido que se reproduzca. La idea es evitar que el resto de las funciones implementen la reproducción del audio, sino que simplemente llamen a una función de este script la cual reproduce un audio concreto, y así si en algún momento se quiere modificar algún audio no es necesario buscar donde se usa sino sencillamente cambiar la función en sí, y ya todas las funciones que la llamen habrían cambiado. Entre las funciones más relevantes del *AudioManager* están reproducir el sonido de los pasos de la protagonista y de los diálogos.

5.4.5. ConversationManager

Desde este script se controlan las conversaciones que establece la protagonista con el resto de los personajes interactuables, como ya se explicó anteriormente son un *InteractiveObject* con una *TalkInteraction* asociada. El cambio es que al interactuar con estos objetos la función que se reproduce es mostrar un panel de diálogos en el que habrá una lista de opciones que el jugador o jugadora puede pulsar, cada opción es uno de los posibles diálogos que la protagonista le puede decir a los NPCs.

En este *manager* la función más importante es mostrar un tipo de dato de diálogo que se le pase como argumento, de tal forma que desde el *TalkInteraction* se le pase los datos de diálogos asociados a cada personaje y el *ConversationManager* los mostrará por pantalla y gestionará las posibles respuestas o subdiálogos. Al construir la conversación en pantalla el ultimo botón siempre servirá para volver al nivel anterior del árbol de diálogos y si está en el nivel más alto para salir de la conversación. Para entrar a un subárbol de diálogos hay datos de diálogo que apuntan a una nueva lista, si este apuntador es nulo o vacío significa que no hay subdiálogo, si contiene información entonces se reconstruirá la interfaz de diálogo para mostrarle al jugador ese subárbol. Con el ejemplo de las figuras de a continuación se entiende mejor cómo funciona.

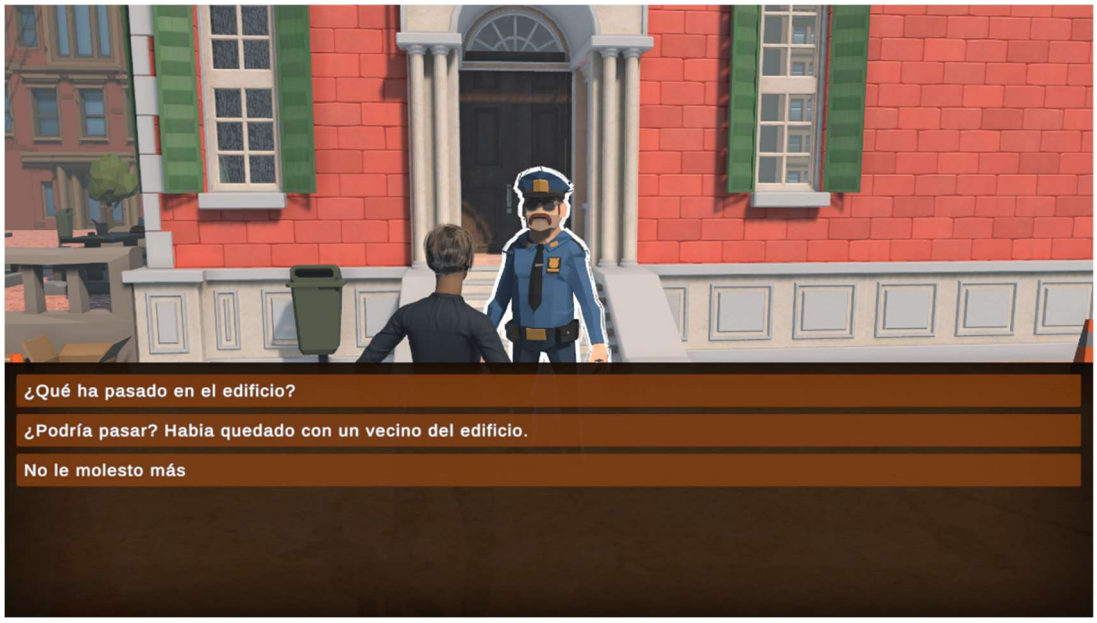

Figura 13: Primer nivel del árbol de conversación con el policía

Figura 14: Segundo nivel del árbol de conversación con el policía

Como se ve la Figura 13 es el primer nivel del árbol de diálogos de uno de los personajes de la escena inicial, si el jugador selecciona la primera opción preguntando “qué ha pasado en el edificio” el NPC le contestaría informándole de que “ha ocurrido un asesinato”, y en ese momento se mostraría el siguiente nivel de esa rama en la que se le puede preguntar más información una vez el policía ya ha informado de lo ocurrido.

5.4.6. MenuManager

Encargado de gestionar la funcionalidad asociada al menú de pausa dentro del juego. A través de este menú el jugador podrá realizar varias acciones que se ven en la siguiente Figura:

Figura 15: Menú principal

La función de este *manager* es mostrar y ocultar el panel que da acceso a todas las opciones que se ven en la Figura 21. Esto significa que este script contendrá todos los eventos que se llamarán pulsando los diferentes botones del panel. Guardar/Cargar Partida y opciones abren un nuevo panel gestionado con sus *managers* que se explican a continuación. El resto de los botones cargan la escena del menú principal o salen de la aplicación respectivamente.

5.4.7. SaveLoadManager

Esta funcionalidad gestiona los *slots* de carga y guardado del juego. La función principal son el evento asociado a los botones de carga de cada *slot*, en total hay 9 espacios para que el jugador o jugadora carguen/guarden partida. Cuando se pulsa el botón y se llama al evento se pasa como parámetro el índice del *slot*, que puede ser de cero a ocho. Este evento comprueba si el panel mostrado es el de carga o descarga para saber cual es su función, tras lo cual realiza una llamada a *ZSerialize* para cargar o guardar todos los *GameObjects* que se han asignado que se deben guardar. *ZSerialize* es un paquete de Unity creado por un tercero el cual permite serializar objetos creados desde Unity y sus componentes para así facilitar el trabajo.

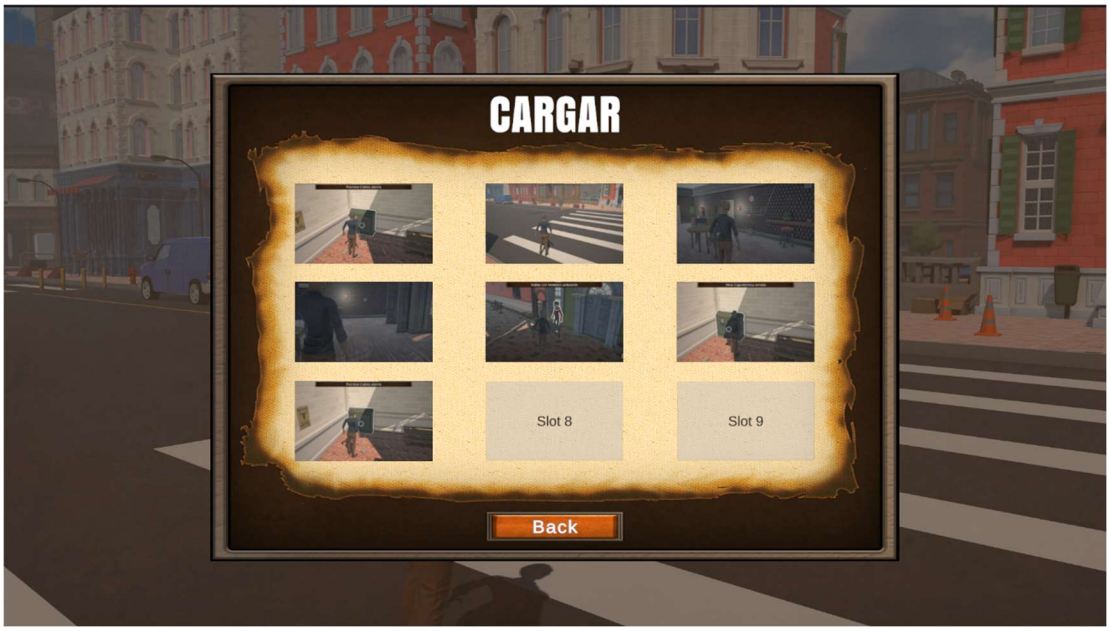

Figura 16: Panel de Carga

Figura 17: Panel de Guardado

5.4.8. SettingsManager

Este comportamiento se encarga de gestionar las opciones seleccionadas por el usuario en el juego, desde el menú principal su función es permitir guardar de manera persistente las variables que el usuario seleccione desde el panel de opciones. Desde el juego tiene doble funcionalidad, aplicar estos ajustes nada más cargar la escena del juego y tener la capacidad desde el menú de pausa y accediendo al panel de opciones guardar las variables que se vayan editando y aplicarlas en tiempo real.

5.4.9. CutsceneManager

Es la función encargada de controlar las animaciones creadas para visualizar a lo largo de los niveles. En el caso del primer nivel hay una animación inicial para poner en contexto al jugador o jugadora de que está pasando. La funcionalidad de comenzar una secuencia de Unity se invoca desde el *GameManager*

5.5. Diálogos y voces

//TODO hasta el final de 5.5

Para el proyecto se ha creado un tipo de dato para representar los diálogos que reproduzcan la protagonista y los NPCs. Estos diálogos se reproducen desde el *manager* de los diálogos como ya se ha explicado, pero el tipo de dato es esencial para el desarrollo de juego. Este tipo de dato esta constituido de una variable texto que será la que represente de manera escrita lo que dice el personaje

dentro del juego y un clip de audio en el que está grabada la voz de los personajes diciendo los diálogos para darle mayor vida al mundo creado. Se ha querido dar especial énfasis a los diálogos porque son de donde nacen las aventuras gráficas y cualquier juego que quiera dar importancia a la narrativa, ya que sirven como medio para que el jugador o jugadora se enteren de lo que pasa en el mundo e incluso empatice con ciertos personajes logrando vincular al juego con su usuario.

Para la grabación de los diálogos se ha buscado a diferentes personas y aplicado para algunos personajes un modulador de voz para que se diferencien de otros. De esta forma se han conseguido estos clips de audio que luego se han importado a Unity y usando la funcionalidad del *manager* de diálogos se reproduzca siempre que así lo requiera la lógica del juego.

5.6. Items e Inventario

Para este propósito es necesario, dentro del *Canvas* ya explicado, crear un panel que contendrá todas las imágenes de los ítems que la protagonista haya ido recogiendo por la escena. Este panel se le asocia un comportamiento que será el encargado de gestionar todo el inventario. Desde el gestor del inventario se puede llamar a un método que añade objetos al inventario y se le debe pasar como parámetro un *Item*, este método se invocará desde la *CatchInteraction* cuando el jugador o jugador coja un objeto para así añadirlo a su inventario. Luego dispondrán de otros métodos como borrar un *Item* del inventario, para cuando ya se haya hecho todo el uso se borrará automáticamente del inventario.

Los ítems se crean dentro del panel de inventario y estarán colocados en forma de malla, tendrán asociados un nombre, un texto por si el jugador quiere mirarlos para obtener alguna pista, la imagen que se muestra dentro del inventario para que sean reconocibles y unos usos. Los usos son una lista de una clase que contiene la información de uso que tiene un *Item*, la lista puede contener de un solo uso hasta todos los que se desee a la hora de diseñar los retos y puzles. Una vez se usa un *Item* se borra de la lista el uso, y cuando esta lista se queda con cero elementos el *Item* se borra del inventario. La clase contiene el nombre del objeto interactivo con el que va a interactuar el *Item*, el resultado del uso que puede ser otro *Item*, una funcionalidad o ambas y un dialogo que dirá la protagonista tras usar el *Item* de manera correcta para que el jugador sepa lo que ha hecho. Estos ítems se podrán combinar con objetos de la escena o con otros ítems del inventario, al almacenar el nombre del objeto con el que se usa cada uno también hay que indicarle si ese nombre hace referencia a un InteractiveObject de la escena u a otro item del inventario.

Para usar un item con un *InteractiveObject* el jugador o jugadora debe estar en el mundo del juego cerca de el para que se establezca como su objetivo y abrir el inventario y darle clic al item que quiere usar sobre el objeto, de esta forma se comprueba si el tipo de uso y el nombre coinciden con los usos del item para realizar la funcionalidad resultante de la combinación, que puede ser un nuevo item, una

funcionalidad o ambas. Para el combinar ítems sigue una lógica similar, pero teniendo en cuenta que cuando el jugador está en el inventario también puede asignar un objetivo, para ello se permite que cada elemento del inventario, las imágenes que representan los ítems, tenga un comportamiento asociado de arrastrar y soltar. Cuando se detecte que el jugador o jugadora comienza a arrastrar uno de estos elementos se marca como el actual objetivo dentro del gestor del inventario y cuando suelte el *item*, en caso de que el cursor este sobre otro se procede a comprobar si tiene entre sus usos el nombre del objeto que se está intentando combinar. El resultado de combinar ítems normalmente es otro *item* pero existe la posibilidad de que también ejecute alguna funcionalidad, como ya se explicó en las funcionalidades asociadas.

5.7. Puzles

Se ha creado un *prefab* base y un comportamiento asociado a los puzles exógenos que se propongan en las aventuras. Esta funcionalidad incluye la existencia de un *Canvas* que contiene los elementos visuales del puzle y dos botones, uno que permita al jugador o jugadora salir del puzle y continuar con la aventura y otro que reinicie el puzle por si el usuario prefiere volver al estado inicial.

Para entrar en el puzle es necesario que se oculten el resto de los elementos de la IU que sobren y se comience a mostrar el que contiene el puzle, a la vez que se bloquea las interacciones y movimiento de la protagonista. De esta manera se busca que el jugador focalice toda su atención en el reto que el juego le propone hasta que decida salir o finalice el puzle con éxito. Los puzles deben tener un *FuncionalityObject* como ya se ha explicado anteriormente, desde el comportamiento del puzle se detecta cuando se complete con éxito el puzle para cerrar el panel gráfico del puzle y ejecutar la funcionalidad, que puede ser abrir una puerta, una caja o desbloquear nuevas funcionalidades en nuevos objetos.

Ahora podemos hablar del puzle en concreto que se ha implementado para este primer nivel. Para la implementación de este puzle se ha seguido uno ya existente llamado “Klotski”, originario del siglo 20, que consiste en mover unas piezas que aparentemente están encajadas para sacar una de ellas, que estaba inicialmente bloqueada, por un hueco en el otro lado.

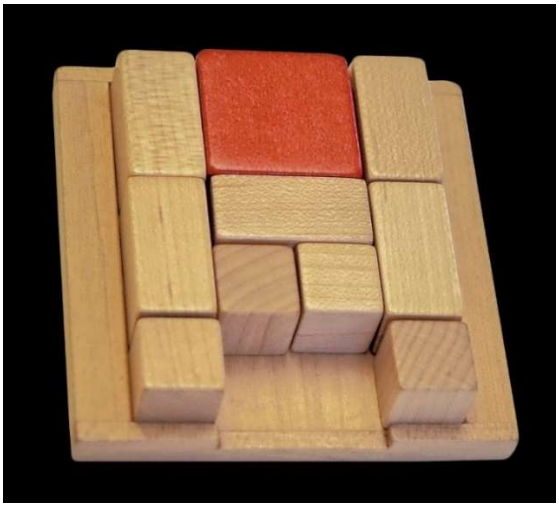

Figura 18: Puzle “Klotski” original

Para la implementación de este puzle se ha usado la base ya explicada anteriormente para tener un marco sobre el que construir el puzle. Una vez se tenga la base, se ha construido el puzle, compuesto por: los marcos de la zona en la que se desplazan las piezas, y las piezas colocadas en el interior en el orden adecuado para construir el puzle y así ajustar su dificultad. La idea es crear un nuevo comportamiento para la pieza que cuando el jugador la arrastre en vertical u horizontal se mueva si es que tiene hueco, y añadir una bandera que controle si esa ficha es el jugador o no. En caso de que no sea el jugador nunca podrá atravesar la zona que detecta que el puzle se ha completado. El hueco por el que debe salir la pieza principal, además de permitir a la ficha del jugador lo atraviese debe indicar que el puzle se ha completado con éxito.

Figura 19: Puzle Klotski dentro del juego

5.8. Requisitos de instalación

Para jugar es necesario un ordenador con Windows, ya que la *build* que se obtiene de Unity es un *.exe (executable)*. En principio estos son los únicos requisitos, aunque es muy recomendable tener un ordenador con una gráfica dedicada para que el juego vaya todo lo fluido que es deseable.

También se ha probado sobre una Steam Deck usando *Proton*, que es una capa de compatibilidad, que está desarrollando Valve, para permitir jugar juegos de Windows en Linux. Aunque es cierto que da algunos problemas con la resolución, el juego se abre y se puede jugar si usamos los *trackpad* con los que cuenta este ordenador de mano/consola, y mapeamos para que el joystick funcione como las teclas de movimiento y los botones del mando integrado sean las teclas de interacción.

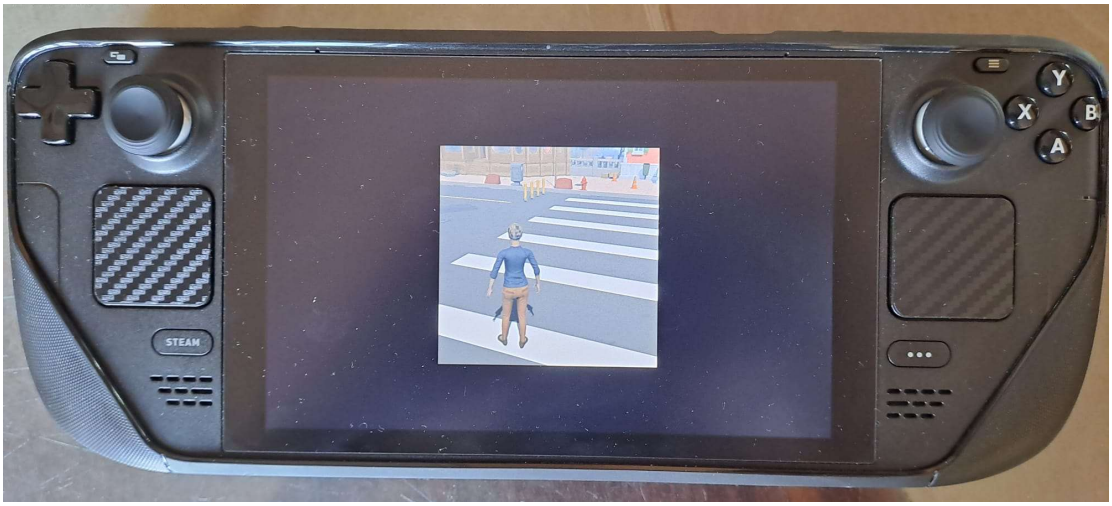

Figura 20: Juego funcionando en la Steam Deck

Además de permitirse jugar en este tipo de dispositivos para el desarrollo de juego se ha utilizado el *New Input System* añadido en Unity recientemente. Esto permite mapear todos los controles creados para el teclado y ratón en un mando y de esta forma dar la opción a los jugadores y jugadoras usar el mando para manejar el personaje e interaccionar con todos los objetos.

5.9. Instrucciones de instalación

El juego al buildearlo en Unity genera una carpeta que contiene toda la información y datos del juego y un .exe que se puede ejecutar desde un ordenador con Windows y nos lanza la aplicación. La carpeta del juego que crea Unity con la aplicación contiene varios archivos y carpetas, el que interesa para la ejecución del juego es el archivo ejecutable llamado “El Legado Olvidado.exe”. Este archivo al abrirse lanzara el menú principal del juego desde el cual se podrá comenzar a jugar entre otras opciones que se explican en detalle en siguientes puntos.

6. Demostración

6.1. Flujo de pantallas y guía de nevegación

Para la explicación del flujo de pantallas se seguirá el mismo proceso que vería el jugador o jugadora al abrir el proyecto. La primera escena se verá al abrir la aplicación desde el menú principal. A partir de esta escena al jugador se le da la opción de ir al juego de varias maneras diferentes, que se explicarán a continuación, para abrir el panel de opciones y el usuario configure el juego como quiera, o por último salir de la aplicación.

Abrir el menú de opciones muestra un panel de configuraciones y ocultará el panel principal con los botones de la Figura 15. Desde este panel se le da al jugador la posibilidad de cambiar varias opciones como: el audio del juego (si desea ver los subtítulos), la sensibilidad de la cámara, la distancia con la que se remarcaran los *InteractiveObjects* (para ayudarle en caso de que no encuentre ninguno) y por último la calidad de los gráficos (que tiene tres modos: bajo, medio y alto). Además de estas configuraciones el panel le permite volver atrás, se ocultará este panel y volverá a mostrar los botones del menú principal, Figura 15.

Figura 21: Panel de opciones

Para entrar a las zonas jugables desde el menú hay tres opciones diferentes: continuar desde la última partida guardada, cargar una partida guardada en concreto o empezar una nueva partida. Hablando

del orden en el que el jugador o jugadora usaría estos botones de acceso al juego, el primero que usaría al iniciar por primera vez sería el botón de “Nueva Partida”, además de porque es lo más lógico los botones de continuar y cargar partida se desactivan cuando no hay partidas guardadas, y así evitar que el jugador pulse botones que no le permiten realizar ninguna acción. Una vez el jugador accede por primera vez al juego a través de la nueva partida, tras la animación inicial podrá interactuar y moverse por el mundo. Si una vez aquí el jugador o jugadora decide guardar partida y volver al menú principal vería como los botones de continuar y cargar partida ya están activos y listos para ser usados. El botón de “Continuar” llevaría al usuario al mismo punto donde guardo la partida por última vez, mientras que el botón de “Cargar partida” abre un panel con los nueve *slots* de partida que existen, de los cuales los que contienen una partida guardada que se puede seleccionar están rellenos con una imagen del momento en el que el jugador guardo partida como se ve en la siguiente Figura 17. Al pulsar este botón lleva al usuario al estado del juego donde guardó esa partida.

Dentro del juego hay otras pantallas y estado del juego de interés ya nombrados en la implementación, pero se nombrarán de nuevo brevemente. Desde el juego se permite abrir el inventario que bloquearía la interacción de movimiento del personaje principal y mostraría el cursor para poder seleccionar cualquier ítem que haya en el inventario, como se vio en la Figura 12. Otra opción del jugador es acceder al panel de opciones desde donde se permite volver a la escena del menú principal entre otras opciones como ya se explicó en el *MenuManager*. En la Figura 21 se pueden ver todas las opciones que se le da al usuario, para el interés del flujo de estados del juego está el botón de opciones que permite acceder al usuario de nuevo al panel de configuraciones que ya vio en el menú principal. Los otros dos botones serían cargar y guardar partida, los cuales llevan al mismo panel visto en la Figura 16 y 17 respectivamente, a través de los cuales se permite seleccionar un *slot* y guardar partida, si se está en modo de guardado, o seleccionar un *slot* de partida ya existente para cargar el estado del juego cuando se guardó ese *slot*.

6.2. Prototipos

Como ya se explicó el método de desarrollo seguido es SCRUM, el cual requiere de iterar sobre una solución y mediante las pruebas con usuarios obtener la versión final, solucionando errores y añadiendo elementos a partir de los resultados aportados por los jugadores de prueba.

6.2.1. Prototipo de controles básicos

El primer paso para comenzar a crear las funcionalidades, descritas en la implementación, es crear una escena básica de Unity en la que se coloque un panel que sirva de suelo y colocar formas básicas de Unity que sirvan de base para añadir las funcionalidades, para posteriormente sustituir estas formas básicas por los elementos artísticos que se quieran mostrar. Al principio se crea una capsula para

representar al jugador sobre el que se implementó el movimiento por la escena, que es la interacción más básica necesaria para poder probar el resto.

Antes de comenzar con la implementación de los controles básicos de movimiento del avatar ~~de la~~ protagonista y de la cámara se han observado otros referentes que tuvieran unos controles similares. Lo importante de esta fase es no innovar innecesariamente sino usar controles reconocibles por los jugadores para así facilitar el trabajo, no solo de la creación de esta fase sino el de aprender a jugar al título. Esto no significa que se haya copiado tal cual un tipo de control, sino que se han observado varias referencias del mundo de los videojuegos modernas que compartan el tipo de control, como cámara libre en tercera persona, para poder usarlos como referencia. Ente los juegos que se han utilizado para este control son Hogwarts Legacy, Red Dead Redemption 2 y The Last Of Us, entre otros. Logrando una composición en pantalla que se considera adecuada, gracias a dejar aire a la derecha del personaje para permitir colocar el inventario en esa zona de la pantalla cuando se abre o una cámara cercana cuando el jugador o jugadora permanece estático, sin embargo en el momento que comienza a moverse el personaje la cámara se aleja dando una mayor amplitud al campo de visión del mundo del juego.

Figura 22: Composición en pantalla del videojuego Battlefront 2

Sobre esta funcionalidad primigenia se ha ido iterando e intentando pulirla lo máximo posible para obtener la mejor experiencia posible para el usuario.

6.2.2. Prototipo de funcionalidades base

Una vez la capsula del personaje se mueve por la escena se crean cubos para representar un *InteractiveObject* y se implementa su funcionalidad, para cuando el jugador pase cerca se marque ese objeto, acompañado de una barra superior en la interfaz que muestre el nombre del objeto. Tras esto

se implementan los diferentes tipos de interacción ya explicados: hablar, mirar, coger y usar. Y para comprobar que todo funciona se implementa la interfaz gráfica referente al inventario, las conversaciones y los diálogos para que al menos se muestren los subtítulos y probar que todo está funcionando. Habiendo implementado lo más básico ya se tiene el primer prototipo que es funcional y permite ir iterando sobre él para mejorar el juego hasta finalizar todas las mecánicas.

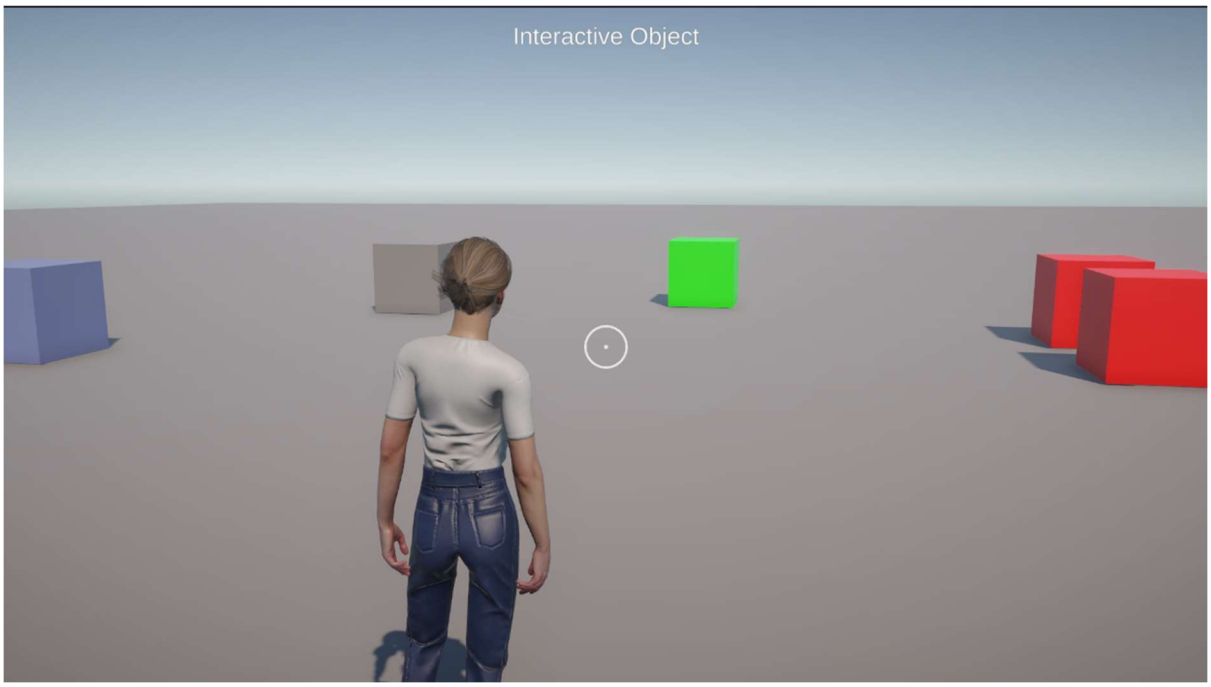

Figura 23: Prototipo del juego

A partir de este prototipo creado se comienza a implementar el arte del nivel y los modelos de los objetos, además de toda la lógica del nivel como que objetos hay que coger, como se deben de utilizar y cual es el orden en el que el jugador debe realizar las acciones. En paralelo se continúan implementando el resto de las funcionalidades ya descritas que son necesarias para obtener un producto final y totalmente jugable con una experiencia de usuario.

6.3. Tests

Al igual que con los prototipos los test o pruebas de usuarios realizadas están relacionadas con la metodología de trabajo escogida al comienzo del proyecto, SCRUM, que ya se ha definido. Estas pruebas se han realizado de manera iterativa sobre el prototipo para adquirir nuevas ideas que pueden ser interesantes a tener en cuenta a lo largo del desarrollo, y así ir ajustando todos los parámetros referentes a la jugabilidad, para mejorar el producto final.

A continuación, se habla de en que punto del desarrollo se han realizado los test, teniendo en cuenta que cada vez que se ha realizado pruebas con usuarios se ha seleccionado 5 personas con diferentes rasgos sociales. Este número se recomienda en estudios de ingeniería del software porque se considera suficiente para poder descubrir errores y detalles que durante el desarrollo pasen por alto, pero no es un número excesivo que pueda complicar la recopilación de la información obtenida con los test.

El primer test se realiza sobre el prototipo explicado anteriormente, esta prueba es importante para obtener información sobre las mecánicas básicas, si son las adecuadas y si la idea principal es jugable y puede entretener a jugadores y jugadoras una vez esté finalizada. De esta prueba se obtuvo información muy relevante sobre la colocación de la cámara, la velocidad del personaje o la ubicación de los elementos visuales. Aunque estos puntos puedan parecer menos importantes es necesario prestar mucha atención a ellos en lo que a *game design* y *level design* se refiere puesto que es la base sobre la que se cimentarán el resto de las mecánicas y funciones. Aunque no se busca un público masivo puesto que el propio genero del juego reduce mucho sus posibilidades, al no ser excesivamente comercial y la gente considerarlo un género que se ha perdido a lo largo del tiempo, darle un aire nuevo con controles y jugabilidad más moderna es interesante, y puede aportar aire fresco dentro de este tipo de juegos, siempre refiriéndose a la pequeña base creada. Además, con todo esto se obtuvo un resultado positivo del camino del juego.

La segunda prueba con usuarios se realiza con una versión inicial del nivel que se ha implementado, colocando ya los *assets* que se iban a utilizar mejorando toda la jugabilidad, hasta tal punto que se reconstruyó el movimiento del personaje por completo, e introduciendo nuevas mecánicas. También se añadió sobre toda la funcionalidad creada los primeros retos que se iban a proponer a los jugadores. En este punto los jugadores dieron información de interés de cara a la interacción con las mecánicas implementadas. Hasta ese punto los controles para interactuar con los ítems en el inventario requerían de demasiada explicación y complicación de cara a los jugadores y las jugadoras, durante las pruebas reportaron que era necesario simplificarlo. También detectaron problemas con la cámara y su colisión con los elementos del escenario y ciertos fallos con las interacciones de los objetos de la aventura. Tras esta prueba se solucionaron todos los errores encontrados y se tienen en cuenta los comentarios de mejoras que se puedan añadir.

El último test se realiza sobre una versión prácticamente final, con todos los retos del nivel finalizado, la interfaz gráfica con su aspecto final, los parámetros de jugabilidad ajustados y siguiendo toda la información recabada en las anteriores pruebas.

7. Conclusiones y líneas de futuro

7.1. Conclusiones

Tras realizar este primer gran proyecto en Unity se han sacado muchas conclusiones positivas. Pese a que es la primera vez que se ha realizado un proyecto grande en Unity y ha habido momentos en los que gestionar un proyecto tan grande ha sido complicado, pero se han sacado un aprendizaje de cara a futuros proyecto o a finalizar este mismo.

7.1.1. Aprendizaje

El primer punto importante a la hora de abordar un proyecto de mayor magnitud es ser consciente del alcance al que va a llegar el proyecto y la limitación que existe al ser una única persona la que se encarga de realizar todos los campos dentro de él. Pese que durante el proyecto se han usado *assets* de terceros ya creados, estos dan la base para la creación de niveles, pero el diseño de los mismos y la colocación de todos los elementos que lo componen sigue cargando sobre la única persona dedicada al proyecto. Esto sumado al diseño e implementación de las interfaces gráficas y, lo más importante, toda la lógica del juego hace que un proyecto aparentemente razonable en cuanto a tamaño de producción, para una sola persona puede convertirse en algo que requiera excesivo trabajo.

Otro punto a tener en cuenta al realizar estos proyectos *indie* en solitario es la cantidad de tiempo que se le puede dedicar, no es lo mismo trabajar en ello a tiempo completo, parcial o como *hobbie*. Esto no solo significa estirar la duración de un proyecto, sino que también requiere una fuerza de voluntad del desarrollador para evitar abandonarlo debido a alargarse excesivamente en el tiempo.

A nivel programación se han utilizado ciertos patrones que al realizar juegos más pequeños con menor número de funcionalidades y más simples no son tan necesarios. Lo más importante y que se ha intentado aplicar al máximo en el proyecto ha sido la encapsulación de las funcionalidades, en los *managers* por ejemplo. La idea ha sido que para cada mecánica diferente que se ha implementado, como lo son el movimiento, la interacción con los objetos de la escena, el funcionamiento de esos objetos y la gestión del inventario, entre otros, se cree un script que almacene esa función. De esta manera a ojos de la interacción del jugador coger un objeto será llamar una función de ese objeto interactuable, siendo así mucho más limpio el código y así más fácil de localizar errores o de gestionar toda la información necesaria. En definitiva, la encapsulación por funcionamientos en videojuegos es totalmente necesaria para conseguir un producto final óptimo.

Otro punto importante en videojuegos es seguir el paradigma de la programación orientada a eventos, esto quiere decir que hay que evitar a toda costa el sondeo de preguntar continuamente, sino que la ejecución de las funcionalidades debe ir precedida siempre de un evento. Los videojuegos son mundos

virtuales en los que puede haber implementadas funciones que están continuamente ejecutándose para dotar de la sensación de vida, pero la gran mayoría de acciones vienen precedidas por un evento del jugador, ya sea un clic en un botón, un movimiento de un joystick o ratón o cualquier otro tipo de entrada de la que disponga y sea compatible con el juego. Esto permite obtener un producto mejorado y se ha tratado de aplicar a lo largo del desarrollo lo máximo que se ha podido tratando de que prácticamente todas las funcionalidades partieran desde una clase llamada *PlayerInteractions*, a partir de la cual se propagaban los eventos que producía el usuario hasta los objetos con los que interactuaba. Esta es una práctica que no siempre se ha seguido en anteriores proyectos de menor magnitud por parecer más cómodo preguntar en cada *frame* por ciertas cosas, pero a la larga resulta mucho más eficiente y fácil de depurar el hecho de que los eventos se propaguen, para que así las funciones finales tengan una pila de llamadas lógicas en la que todo se propaga desde los inputs del jugador o jugadora.

Otro punto que se ha utilizado y ha servido para un aprendizaje en este ámbito es el tratamiento de las herencias. Durante el proyecto ha sido necesario usar la herencia para implementar las interacciones, debido a que muchos objetos de la escena debían de permitir que se interactuaran con ellos, pero cada uno debía de actuar de manera diferente y la manera más eficiente era crear una base común en todos los objetos interactivos pero que cada uno tuviera su propia implementación en función de si se quería coger, hablar, mirar o usar. Es cierto que a lo largo del tiempo que ha durado el desarrollo gracias a otras asignaturas cursadas en el master se ha aprendido que la manera más correcta hubiera sido crear una interfaz y que las implementaciones de la misma escribieran la funcionalidad, puesto que el evento de interacción debía mantenerse en todas las implementaciones de la interfaz, pero finalmente debido a la falta de tiempo no ha sido posible hacerlo de esta manera, aunque al fin y al cabo lo enriquecedor de este proyecto no es solo lo que se ha logrado realizar sino ser consciente de cuales son las limitaciones y que elementos son mejorables y de que forma para poder aprender de cara al futuro.

En conclusión, el mayor aprendizaje ha sido dentro del ámbito de la programación y el desarrollo puesto que, aunque el juego cuente con un apartado visual, el objetivo desde un principio era potenciar habilidades de programación e ingeniería del software para implementar unas mecánicas complejas y divertidas, a la vez que acordes al tipo de juego que se ha propuesto.

7.1.2. Objetivos propuestos y cumplidos

En cuanto a los objetivos desde un inicio se estableció una línea obligatoria y de mínimos de implementar un proyecto de cara a poder ser jugable y tener las mecánicas básicas del tipo de juego que se ha propuesto. Aun con esto es cierto que ha sido imposible llegar a todos los objetivos propuestos, especialmente teniendo en cuenta lo explicado en el aprendizaje. Ciertos objetivos iniciales eran inviables teniendo en cuenta de que se trata de un proyecto que se ha desarrollado

durante cuatro meses a la par que tres asignaturas del máster. A continuación, se habla más en detalle de qué se ha cumplido y qué no.

A nivel mecánico se han conseguido todos los objetivos, ya explicados. Se ha conseguido un control fluido para el jugador o jugadora, así como demás interacciones con objetos y personajes de la escena y crear un sistema de inventario para implementar todos los retos. También se ha implementado un puzle basándose en un antiguo acertijo llamado “Klostki”, como se ha explicado en la implementación de los puzles. La clave de todos los objetivos conseguidos no es que se haya implementado un nivel en concreto y que sería necesario volver a implementar todo para el resto de los niveles, sino que haciendo uso de todo lo aprendido y conocimientos previos se ha creado una base para facilitar la implementación de las mecánicas en futuros niveles. La idea era crear una base para todos los tipos de objetos que existen en la aventura e ir usando los objetos funcionales que permiten ir innovando e introducir objetos diferentes al gusto del desarrollador. De esta forma el proyecto creado no solo es el nivel de un juego sino la base para facilitar la creación del resto del juego, por ello se ha dado mucha importancia a que sea una base sólida y consistente, siendo uno de los principales focos de atención del desarrollo la depuración y solución de cualquier error que pudiera producirse en diferentes situaciones.

Aunque también es cierto que ciertas mecánicas no han dado tiempo de implementarse. Inicialmente se habló de introducir una mecánica de sigilo, que al final por falta de tiempo no se ha implementado, esto sumado a que en el primer nivel esta mecánica carecía de sentido puesto que no existe ninguna zona en la que sea lógica utilizarla. Pese a esto como se comenta más adelante en líneas futuras, no significa que continuar el desarrollo del juego vaya a implicar no crearlas, sigue siendo una mecánica interesante y que puede ser muy útil incluso para combinarla en ocasiones con retos de inventario para poder continuar.

Otra idea inicial era llegar a mostrar el segundo nivel donde la historia se podría desarrollar y así ver como es una escena más cerrada y reducida, pero el problema de nuevo es la falta de tiempo y la magnitud de un proyecto así hecho por una sola persona. Aunque se han presentado ya todas las mecánicas del juego en este primer nivel, quitando el sigilo como ya se ha explicado, el proceso para crear los siguientes niveles ~~es~~ siguiendo la historia, explicada en la propuesta, es pensar o idear un nivel, donde se va a desarrollar y cuales son los retos y puzles que se le quieren proponer al jugador o jugadora. El segundo nivel está planeado que fuera en el apartamento y el portal del edificio donde se entra al acabar la escena final.

7.1.3. Seguimiento de metodología y la planificación

Dentro de la planificación planteada se considera que el seguimiento ha sido óptimo teniendo en cuenta la inexperiencia en la planificación de desarrollo de videojuegos completos. En ocasiones la

planificación no se ha cumplido, pero siempre ha sido para adelantarse a las fechas marcadas. Ciertas funcionalidades e implementaciones que debían hacerse en un tiempo se anticiparon y se comenzó con las siguientes etapas para así adelantar el trabajo lo máximo posible. La metodología pese no ser específica para el desarrollo de videojuegos, sino para *software* en general, ha sido correcta, permitiendo iterar sobre un producto y generar una versión final totalmente jugable, y deseable considerando los recursos. Además, las pruebas con usuarios han ayudado a encontrar varios errores o seguir consejos dados por personas que han jugado otros títulos, estos test a lo largo del desarrollo agilizan el proceso permitiendo cambiar ciertos elementos jugables del juego que no son óptimos de cara al jugador. De esta forma se evita que se conviertan en fallos de la jugabilidad irreparables una vez sea lanzada una versión final.

En conclusión, se considera que la metodología seguida y la planificación creada sobre ella ha sido la apropiada para un desarrollo de estas características, y tanto es así que se ha conseguido cumplir todos los plazos de manera correcta y, aunque no se haya llegado hacer todo lo propuesto inicialmente, si se ha obtenido un producto final y funcional.

7.2. Líneas de futuro

Como ya se ha dicho el objetivo del proyecto no es abandonarlo tras la entrega del trabajo final de máster. A lo largo del proyecto se ha escrito una historia completa para el juego, una pequeña línea a seguir sobre que niveles debe contener la historia. Se han diseñado las mecánicas de juego y como el jugador debe utilizarlas, además de implementar toda la base lógica de ellas para que desde Unity sea tan cómodo como arrastrar estos nuevos *GameObjects*, colocarlos en la escena y añadirle su modelo y los diálogos, ítems o funcionalidades asociados a ellos de manera concreta. Esto facilitara el final del desarrollo ya que solo es necesario preocuparse por los recursos artísticos y el diseño de los niveles de la aventura.

En definitiva, el objetivo principal es finalizar el juego para contar con un producto final y completo que cuente toda la historia, para de esta forma poder contar con un juego finalizado y completo entre los proyectos académicos y a su vez personales.

Bibliografía

Anderson, C. (24 de abril de 2012). *The Man Who Makes the Future: Wired Icon Marc Andreessen.*
Manovich, L. (2011). *The Language of New Media .* Cambridge: MIT Press.
WIRED website: http://www.wired.com/epicenter/2012/04/ff_andreessen/, consultado 12/12/2012

Cladera, R. (2019). *Análisis FODA .* Cladera, https://cladera.org/foda/foda_details.php?id_subcategory=72, consultado 16/06/2023

Devuego: https://www.devuego.es/gamerdic/combo/perfiles-gamers/#:~:text=Jugador%20con%20gran%20dedicación%20e%20interés%20por%20los%20videojuegos%20que,y%20obtener%20las%20máximas%20puntuaciones, consultado 16/06/2023

French, J. (2022). *How to capture the screen in Unity (3 methods).* GameDevBeginner, https://gamedevbeginner.com/how-to-capture-the-screen-in-unity-3-methods/#screen_capture_class, consultado 16/06/2023

GitHub: https://gist.github.com/mattatz/1c039dce5ef251dcef7b628c878bea32, consultado 16/06/2023

HektorDocs: https://docs.hektorprofe.net/escueladevideojuegos/articulos/fases-del-desarrollo-de-videojuegos/, consultado 16/06/2023

Indiefence: https://indiefence.miguelrfervenza.com/2012/06/tipos-de-rompecabezas/#clasificacion, consultado 16/06/2023

Raffin, L. (2012). *Creador de Journey cree que los juegos no son lo suficientemente adultos.* Tecnogaming, https://tecnogaming.com/creador-de-journey-cree-que-los-juegos-no-son-lo-suficientemente-adultos/, consultado 16/06/2023

Redaccion Antevenio. (2021). Perfil del consumidor de videojuegos: así son los gamers españoles. Antevenio, https://www.antevenio.com/blog/2021/01/consumidor-de-videojuegos-espanol/, consultado 16/06/2023

Skip Intro: https://guiondevideojuegos.com/diseno/consejos-disenar-aventuras-gráficas/, consultado 16/06/2023

Squared: https://ceoindie.me/2018/03/18/introduccion-metogologia-scrum-desarrollo-videojuegos-1/, consultado 16/06/2023

Stack Overflow: https://stackoverflow.com/questions/42043017/check-if-ui-elements-recttransform-are-overlapping, consultado 16/06/2023

Talent.com: https://es.talent.com/salary?job=programador, consultado 16/06/2023

Uadedigital: https://uadedigital.files.wordpress.com/2014/08/gamificacion2.pdf, consultado 16/06/2023

Unity: https://forum.unity.com/threads/generating-sprites-dynamically-from-png-or-jpeg-files-in-c.343735/, consultado 16/06/2023

YouTube: https://www.youtube.com/watch?v=fR8txbe2wVg&list=RDCMUC0UD2KSZESc5PCPzTBZC_dw&start_radio=1&rv=fR8txbe2wVg, consultado 16/06/2023

Recursos y assets

Sketchfab: City Park at Sunset - Download Free 3D model by Virginia Vidonis, consultado 16/06/2023

Unity Asset Store: Building Apartment | 3D Environments, consultado 16/06/2023

Unity Asset Store: Cartoon Low Poly City Pack Lite | 3D Urban | Unity Asset Store, consultado 16/06/2023

Unity Asset Store: Dark Brown GUI kit | 2D GUI, consultado 16/06/2023

Unity Asset Store: Footsteps - Essentials | Foley Sound FX, consultado 16/06/2023

Unity Asset Store: Legacy Particle Pack | VFX Particles | Unity Asset Store, consultado 16/06/2023

Unity Asset Store: Modular Stylized Colonial Town | 3D Urban | Unity Asset Store, consultado 16/06/2023

Unity Asset Store: Quick Outline | Particles/Effects, consultado 16/06/2023

Unity Asset Store: Workplace Tools | 3D Industrial | Unity Asset Store, consultado 16/06/2023

Unity Asset Store: Yughues Free Ground Materials | 2D Floors | Unity Asset Store, consultado 16/06/2023

Unity Asset Store: ZSerializer - Save & Load System | Utilities Tools, consultado 16/06/2023